NATURAL

Natural

Single Point of Development (SPoD)

Natural Development Server
Version 1.1.3 for OS/390

software AG

This document applies to Natural Development Server Version 1.1.3 for OS/390. Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Table of Contents

Natural Single Point of Development - Overview

Foreword

With the introduction of Natural Single Point of Development (Natural SPoD), Software AG is responding to the trend of using GUI-based tools with their productivity-enhancing features. Natural SPoD more than satisfies the trend by providing a state-of-the-art development workstation for building, testing and maintaining Natural applications throughout their life cycle across all Natural-supported platforms. The major components of Natural SPoD are:

- **Natural Studio** as remote development workstation (Natural Studio is delivered as part of Natural 5.1 for Windows)
- one or more **Natural Development Servers** which provide access to the Natural system files required for single point of development. A Natural Development Server can be located either on a mainframe computer (currently supported) or on a middleware system (in preparation).

Contents

This documentation covers the following topics:

Single Point of Development (SPoD) Documentation

Introducing Natural Single Point of Development - Overview

Introducing Natural Single Point of Development - Overview

This document covers the following topics:

Introducing Natural Single Point of Development

- What is Natural SPoD?
- Remote Development Interface
- Application Concept
- System Architecture
- System Administration
- Natural-SPoD-Related Frequently Asked Questions

What is Natural Single Point of Development?

The following topics are covered:

- Features and Benefits
- Outlook - One Natural for All

If you have been developing applications using Natural in a non-graphical environment, you have undoubtedly become familiar with Natural's wide range of capabilities and most likely also with some of its inconveniences and shortcomings. You have probably quite often wished for a more advanced user interface for "your Natural". With Natural Single Point of Development, your wish has indeed become reality.

Simply speaking, one might say that Natural Single Point of Development is a Windows-based Natural workstation for remote application development on different platforms.

Actually, Natural Single Point of Development concept covers more than a simple workstation in that it combines the advantages of two disparate worlds:

- the strengths of the character-based Natural products and
- the ease of use provided with Natural for Windows.

In addition, it provides a series of novel features which are explained below.

Features and Benefits

Natural Single Point of Development approach is Software AG's response to the increasing demand for a state-of-the-art development workstation for building, testing and maintaining all Natural applications throughout their life cycle and across all supported platforms.

It meets the following requirements and offers the following advantages:

- **Client/server architecture enabling one single remote development environment for all platforms**

  Natural Single Point of Development is based on a client/server concept. On the client side, Windows-based Natural Studio with its modern look and feel, its powerful drag-and-drop copy and paste functionality and its browser-style workspaces offers one single, uniform view for all Natural users. Its graphical user interface is the basis for a single, uniform working environment for all platforms (mainframe, mid-range and PC computer systems) supported by Natural.

  For more details, refer to System Architecture.

- **Full remote development access after mapping to the target environment**

  For remote development, the Windows-based Natural client (which will remain useable for Windows-only application development) can be enabled to interact with a development server that can be located on a single or multiple, homogeneous or heterogeneous mainframe or mid-range computer systems. Once you have mapped your target environment in the client's browser-style application workspace, you have a set of convenient and effective tools at hand that enable you to perform all development tasks directly in the target development environment. You can use the functionality to the extent it is available there. And to prevent concurrent updates, the object under work on the target platform is locked.

For more details, refer to Remote Development.

- **Single, familiar system image of all objects and ressources involved in application development**

  Natural and non-Natural objects can be accessed and processed with a single user interface.You can generate DCOM components, use RPC and EntireX communications and/or put your applications to the Internet. You can submit and monitor jobs, control job listings and perform dataset maintenance tasks. Operations on all of these different object types, whether they reside on OS/390 (development server already implemented), VSE/ESA, BS/2000, UNIX, Windows NT, Windows 2000 (development servers are under development or planned), is afforded using a well-known graphical user interface.
  Under consideration: You can display and edit text files, JCL, and code held as Natural objects or 3GL programs.

  For more details, refer to First Steps with Natural Single Point of Development.

- **New application concept giving a logical view to distributed applications**

  To meet the requirements of a new, distributed application development scenario, a new-defined application concept has been introduced which provides a logical view of Natural and future objects on the various sites where they are developed and used. On the workstation screen, this is complemented by an application workspace which shows all distributed objects of an application in a tree structure.

  For more details, refer to Application Concept.

- **Remote development server file**

  A new central data dictionary file has been introduced to cope with the requirements of the new, distributed application concept. Having the same structure as the well-known Natural system file FDIC, the new system file serves to store the information where the objects linked to an application are stored and which objects are locked.

- **Reduced training costs plus increased productivity**

  The use of only one easy-to-use working environment and the independence from operating system or TP monitor will reduce the investment in training otherwise required for the different environments and will shorten the turnaround time for application development.

- **Future-proof by pluggable extras**

  As a first example, an XRef GUI Client is available as an optional plug-in unit for the workstation.This facility enables you to retrieve and display conveniently the cross-reference information which is essential for developing applications in Natural. It gives you a comfortable way of listing and navigating through hierarchies of referenced and referencing objects, showing relationships between and within program objects.

  For more details, refer to XRef GUI Client.

  Other plug-ins are under development. Even third-party plug-ins may be easily integrated in the future. Which plug-ins are actually visible and active can be configured on a per user basis using the Plug-In Manager.

- **Enhanced online help/documentation for remote environments**

  For remote environments with character user interface, Natural Single Point of Development enables application developers and administrators to display context-sensitive help and additional documentation in electronic form using browser style windows within the graphical user interface. This means fast and efficient access to vast amounts of information which formerly required many volumes of printed manuals. Should you need complex information in printed form, you can print out your personal copy of each document that is available online.

In addition, the syntax help in the program editor available in Natural Studio offers full-detail context-sensitive help for the following Natural syntax elements:

- Statements
- System variables
- System functions
- Parameters (for example, the AD parameter)

- **Browser-based administration**

  A new browser-based System Management Facility provides the Natural administrator with the ability to easily and efficiently perform various Natural administration tasks. This facility, which is being integrated into all Software AG products, has a web browser style appearance and functionality.

Outlook - One Natural for All

So, is it a Natural developer’s wish come true? The current implementation which supports OS/390 environments is already a decisive step in the right direction. Remote development servers for other platforms and operating systems (BS2000/OSD, UNIX, etc.) are planned or in preparation.

Natural Single Point of Development will then be more than wishful thinking - it will indeed be **one Natural for all**.

Remote Development Interface

The primary goal of the Single Point of Development (SPoD) approach is to provide a development interface that enables software engineers to develop Natural applications for any platform using only the Windows-based graphical user environment of Natural Studio.

The essential new features are:

- **Remote File Manipulation**

  In the Natural Studio views, you can manipulate (e.g., move, copy) program objects, regardless of location.

- **Remote Editing**

  You can retrieve Natural source files transparently from the target environment, edit them at your workstation and then save them to the target environment. The GUI supported dialogs of the editors provide significant advantages over the character-based editors.

- **Remote Compiling**

  You initiate the compilation process from your workstation by issuing commands to the target environment.

- **Remote Debugging**

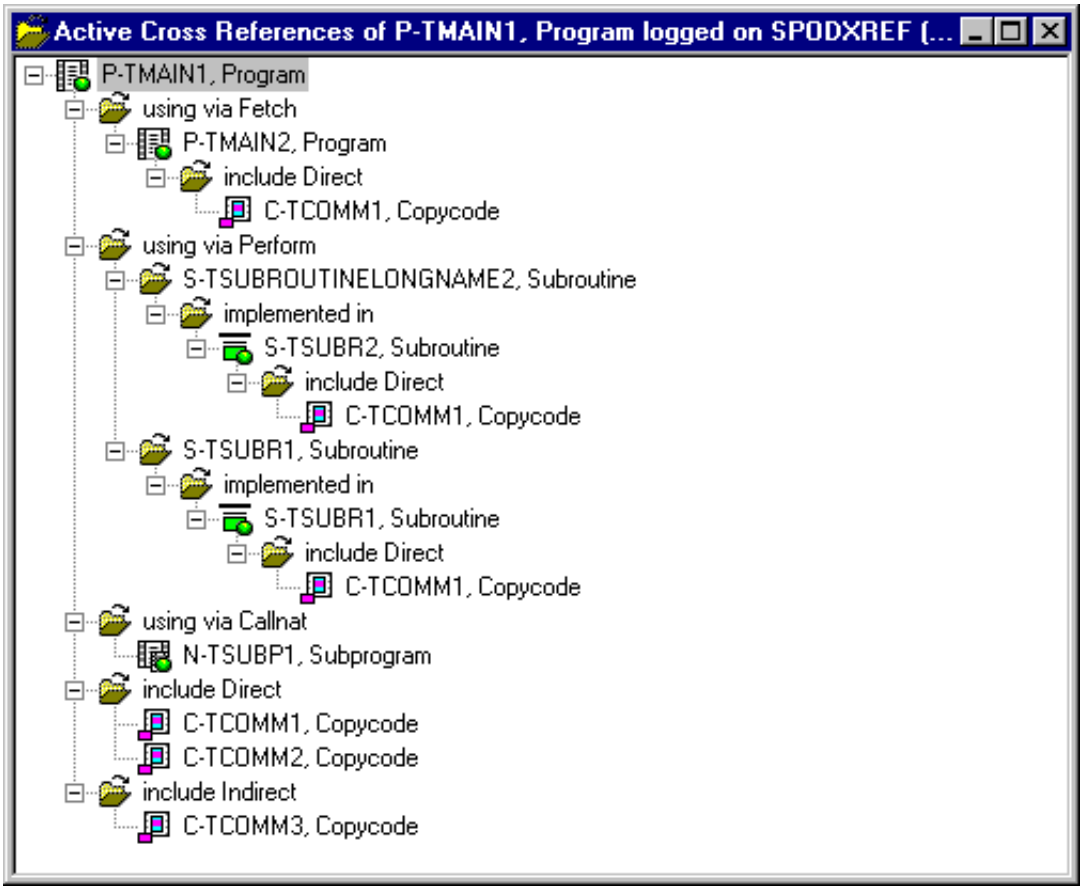
  The debugger available with Natural Studio can be used to debug applications which execute in a target environment that can be located on the same system or in Natural mainframe server environments.

- **XREF Information**

  An XRef GUI client is available as an optional plug-in unit for Natural Studio. It stores cross-reference information created during CAT or STOW commands in a remote development server file (a system file structured like the FDIC system file). This facility enables you to retrieve and display this essential development and maintenance information conveniently in Natural Studio. This gives you a comfortable way of listing and navigating through hierarchies of referenced and referencing objects, showing relationships (cross-references) between and within program objects. See also XRef Data.

  You do not need to have Predict installed on the target environment.

  The figure below shows an XRef example screen. For more details and illustrations, refer to First Steps with Natural Single Point of Development.

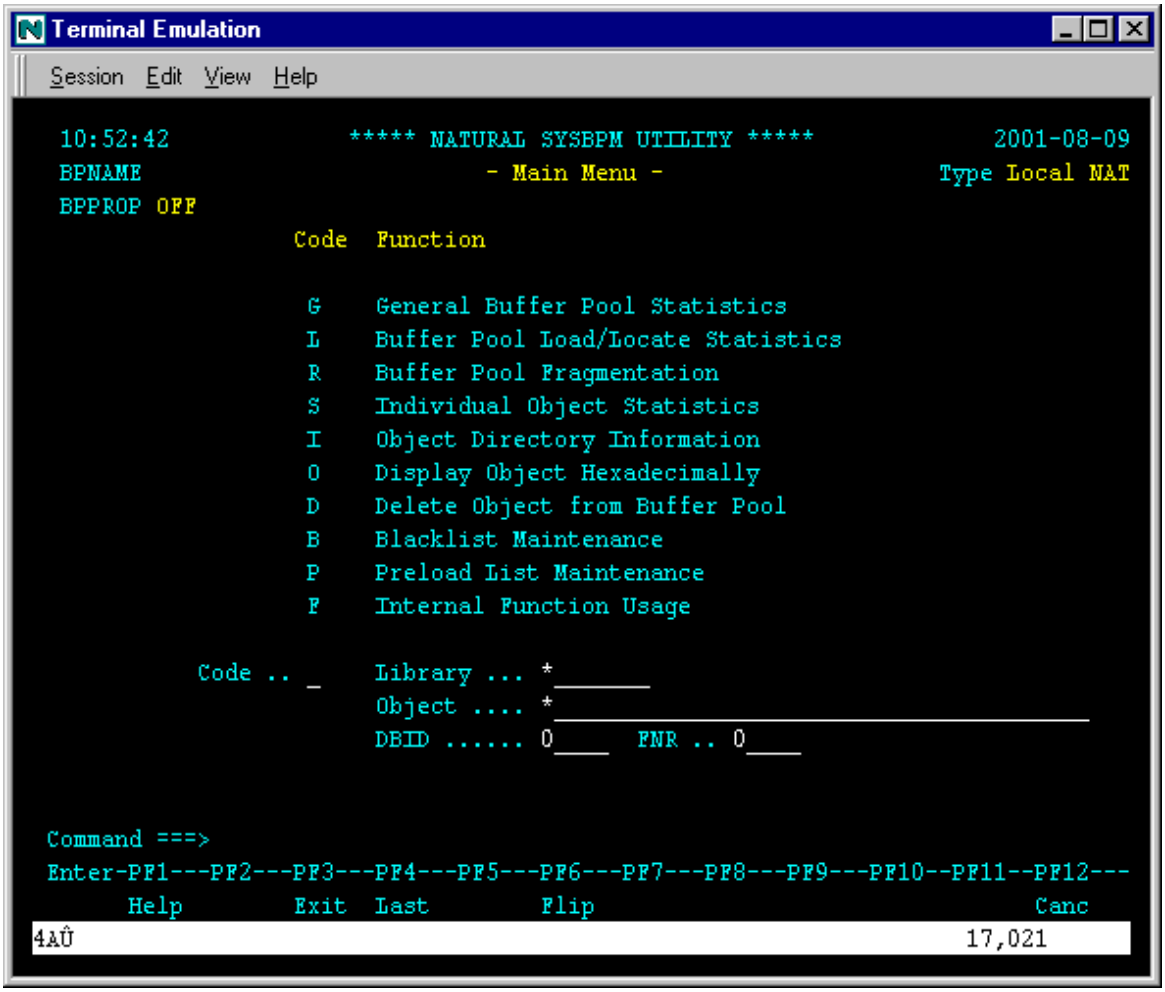


- **Object Locking**

  When you access a remote development server, an object locking mechanism prevents concurrent updates. Locking information is kept in the development server file.

- **Pop-up Terminal Emulation Window**

  Maintenance of mainframe applications often involves testing output to terminals. During remote debugging from Natural Studio, a terminal emulation window pops up automatically for that purpose. This terminal emulation is also available when you are using a utility with a character user interface, for example:



In summary, the benefits of remote development and maintenance are:

- Better control over Natural development tasks
- Increased productivity through a powerful graphical work area
- Lower costs for software development, maintenance and administration
- Greater job satisfaction for software engineers

SPoD Application Concept

The following topics are covered:

- Conventional Approach
- SPoD Approach
- Base Application
- Compound Application
- Application Workspace

Conventional Approach

A conventional Natural application consists of a collection of Natural and Non-Natural objects. Together, they form a functional unit which covers the business logic for a particular business problem. An application consists of a set of libraries and their Natural objects. It is possible that one part of the library belongs to one application while another part belongs to another (different) application.

SPoD Approach

With the SPoD approach, the term "Natural Application" is introduced. It provides a **logical view** of Natural and Non-Natural objects (such as job control files). A Natural application does not contain the objects, but it is a **collection of links** to Natural objects or "sub-applications" describing where the objects are stored. Natural objects or "sub-applications" can be linked to several applications.

In a SPoD scenario, the following types of applications exist:

- base application
- compound application

The space where the applications are maintained and displayed at the workstation is called the Application Workspace.

The definitions are stored in the development server file.

Note: Predict Version 4.2 supports this application concept. Earlier versions of Predict cannot be used.

Some Arguments for Using the Term "Natural Application"

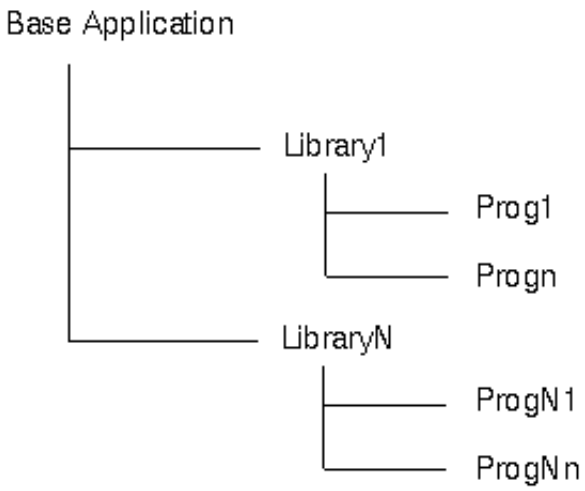
The following topics may give you an idea why you should use the term "Natural Application" and the concept that stands behind it:

- It groups the content of your libraries in a logical order based on the different applications you are maintaining that can be displayed in parallel to the library structure without using an additional tool.
- It focusses your view on the number of objects of a library you are intend to work on. You need no longer navigate through all Natural objects located in one library, because you can edit the objects directly from the Application Workspace.
- It reduces the maintenance effort, because you are able to link all objects of different libraries to the application, especially those located in the STEPLIBs.
- It gives you the opportunity to group an application into different "sub-applications" whereas each "sub-application" can be assigned to a member of the development team that is responsible for its development or maintenance.
- Thinking of client/server architecture, you are able to group your application into different

"sub-applications", and each "sub-application" can be stored on a different platform at application runtime.

Base Application

A base application is defined as a set of Natural object links. The associated Natural objects pertain to one specific Natural Development Server and are all located on the **same** FUSER system file.



Base Application Structure (Example)

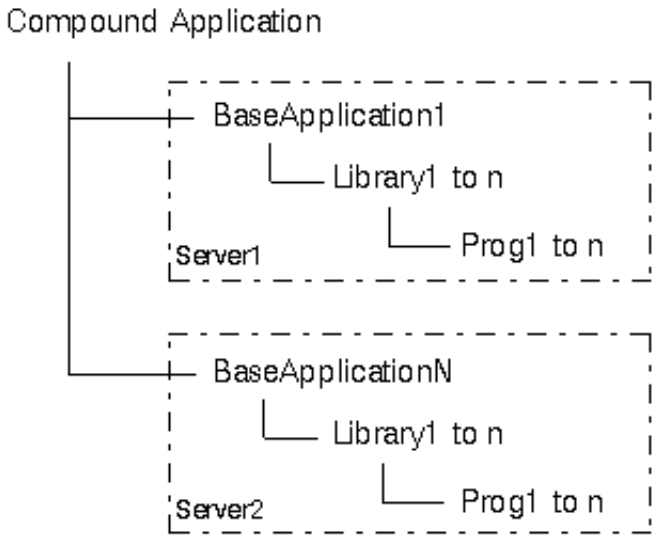
The following information is stored for a base application:

- Name of the application
- Description of the application (textual information only)
- Name and port ID of the Natural development server where the linked objects reside, i.e. the application environment
- Profile for the application environment
- Identification of each object linked

Compound Application

A compound application enables the application developer to combine several base applications. It is defined as a set of links to these base applications.

The base applications involved in a compound application can be spread across **different** FUSER system files or different development servers. Each base application may have a different parameter setting.



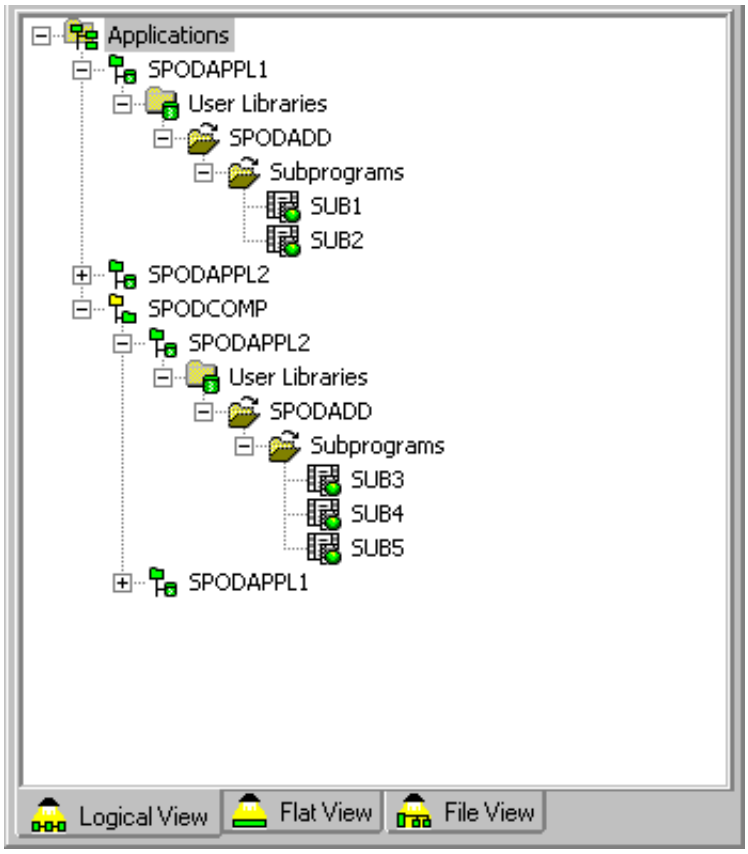
Compound Application Structure (Example)

The following information is stored for a compound application:

- Name of the application
- Description of the application
- Identification of each base application linked

Application Workspace

The Application Workspace is an area on the Natural Studio screen (similar to Natural Studio's Library Workspace) where all known applications and their objects are shown in a view which complements the existing logical, flat and file views and displays a tree structure comprising all program objects belonging to an application, see example below.



System Administration

One aim of the Single Point of Development (SPoD) approach is to provide the Natural administrator with a system management facility with a web browser based appearance and functionality. This facility, the System Management Hub, provides for easier and more efficient handling of all system administration tasks in homogeneous and heterogenous environments.

The System Management Hub can be used for the following tasks:

- display information about the installed Natural products and subproducts,
- rename tree entries.

Information on the properties of the existing Natural installations comprises:

- version number,
- release number,
- system maintenance level number,
- patch level,
- installation date, time and user,
- installation path,
- subproducts (Natural Security, Predict, Super Natural, etc.) with version number and installation date,
- Natural FNAT system file(s).

The System Management Hub also provides a list of all Development Servers and the Windows-based Natural Client. When you select a specific server, the connection between Natural Studio and that server is automatically established. This server will then be visible in Natural Studio’s tree view as an expandable node in a logical or physical view.

SPoD System Architecture

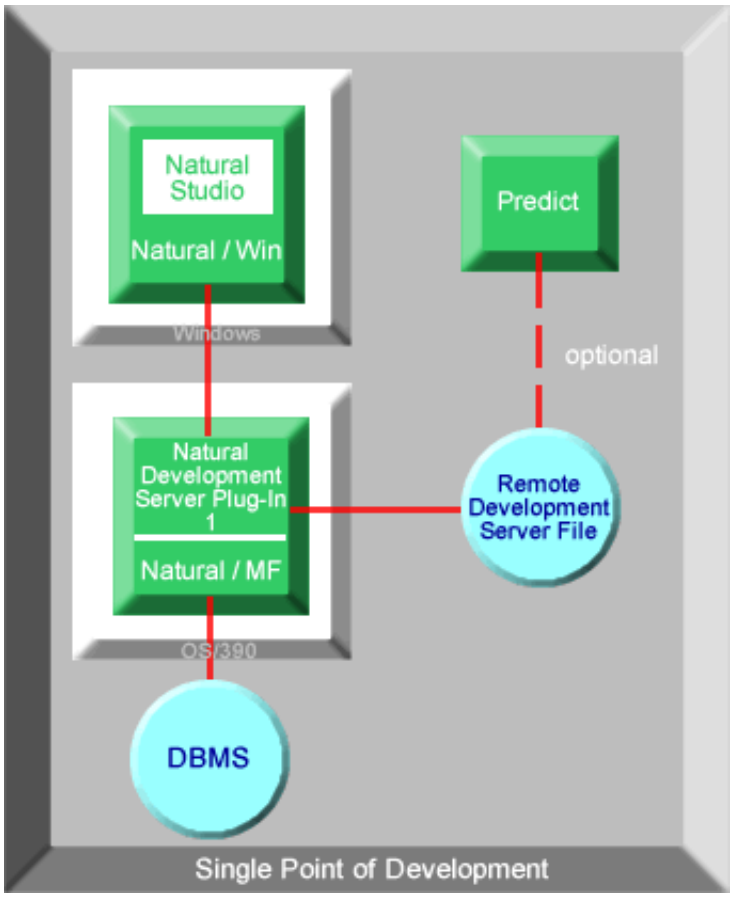
This document describes the Natural Single Point of Development (SPoD) system architecture. The example shown below refers to a remote development server that runs on a mainframe under the operating system OS/390.

The following topics are covered:

- Typical Scenario
- Natural for Windows as Remote Development Client
- Natural Development Server(s)
- Development Server File
- Database Management System
- Predict for Mainframes

Typical Scenario

A typical Natural SPoD scenario on a mainframe system running under OS/390 is made up of the components shown in the following block diagram:



The details of the block diagram are described below.

Natural for Windows as Remote Development Client

As of Version 5.1, Natural for Windows with its graphical user interface can be used as a remote development client. In this case, Natural Studio is the central development workstation for all platforms.

All the Natural-related steps in the application development cycle including the configuration steps can be done from within Natural Studio, independent of the platform where the specific development environment resides. Remote communication with the platform-specific Natural environment takes place via a Development Server located on the different platforms, involving the use of a protocol and using TCP/IP as physical transport layer.

Most of the functionality needed for remote development (administration, application workspace, remote editing, compiling, debugging, etc.) has been integrated in Natural Studio. For storing cross-reference information, you can use a cross-reference GUI client which is available as an optional plug-in unit (not shown in the above figure) for Natural Studio. Additional plug-in units for other purposes are planned.

Natural Development Server

To enable remote development on a specific platform, e.g. on a mainframe system, a Natural Development Server has to be installed. The development server is basically a Natural running in the Natural server environment in conjunction with the add-on product Natural Development Server (NDV) installed.

A Natural Development Server for OS/390 is currently provided. Base requirement is Natural Version 3.1.5 or higher for Mainframes installed. Servers for other platforms are planned. For details, refer to the Natural Development Server documentation.

The remote development server covers the following functions:

- provide access to the system files,
- access the application data,
- use the remote development server file to guarantee the consistency of application modifications by object locking, independent of the workspaces used,
- execute the remote commands issued from the development client (Natural Studio).

Development Server File

A Development Server File stores application data and holds object locking information. This file is structured like the Natural FDIC system file and may coexist with an already existing FDIC file. It cannot be modified dynamically or via user exits. It is addressed using the Natural parameter FDIC.

Database Management System

The Natural remote development environment can be used with any Natural-supported database management system that is available for the specific platform and that runs under the existing operating system.

Predict for Mainframes

Predict can be used in conjunction with Natural SPoD, but it is not required. You can use the XRef GUI Client plug-in for Natural Studio instead. However, if you intend to use your existing Predict installation, it will be necessary that you migrate to Predict Version 4.2 or higher.

Natural SPoD Frequently Asked Questions

This document contains frequently asked questions regarding Natural Single Point of Development. Where applicable, reference will be made to relevant problem descriptions/solutions that are maintained in Software AG's Support Information System SAGSIS.

Which versions of Natural will be useable in a SPoD environment?

As a prerequisite for **creating and operating** a Natural remote development environment, the following versions of the following products are currently required:

- Natural for Windows Version 5.1.1 or higher
- Natural Remote Development Server Version 1.1.1 or higher
- Natural for Mainframes Version 3.1.5 or higher

Any earlier version of Natural for Mainframes or for Windows (and, in the future, Unix or OpenVMS) can be administered in a SPoD environment. The same applies to the applications and objects created with earlier Natural versions. For remote development, however, only the aforementioned Natural versions can be used.

Can Natural objects of previous versions be used in a SPoD environment?

Yes, Natural objects of previous versions can be used in a SPoD environment without migration.

Do I need Predict?

No, although the SPoD approach requires a Development Server File which is a central Natural system file structured like the FDIC file. This does not mean that Predict is a prerequisite for remote development. The SPoD environment features its own XRef functionality which is available as a plug-in. Nevertheless, Predict Version 4.2 and subsequent versions will support the SPoD application concept.

Can I use earlier versions of Predict in conjunction with SPoD?

Definitely not! Using a Predict version lower than 4.2 in conjunction with the remote development environment would destroy the information that is held in the Development Server File. Predict Version 4.2 offers a migration path from the Predict 4.1 FDIC to the Development Server File so that Predict data can be used in Natural SPoD's Application Manager.

Can I use Predict Case in a SPoD environment?

Predict Case (PCA) will not be integrated into the SPoD concept. It can be invoked like any other mainframe Natural application via the terminal emulation feature of the remote development environment. However, Predict Case objects will not appear in the remote development tree view.

Can I use Natural Construct for Mainframes in a SPoD environment?

Natural Construct (CST) for Mainframes will not currently be integrated into the SPoD concept. Currently, it can be invoked like any other mainframe Natural application via the terminal emulation feature of the remote development environment. However, Construct objects will not appear in the remote development tree view. A partial integration of Construct functionality is under consideration.

Can I use Predict Application Control in a SPoD environment?

Predict Application Control (PAC) will not currently be integrated into the SPoD concept. Currently, it can be invoked like any other mainframe Natural application via the terminal emulation feature of the remote development environment. However, PAC objects will not appear in the remote development tree view. A partial integration of PAC functionality is currently under consideration.

Can I proceed without using a SPoD environment?

Yes, you can use the new SPoD-enabled versions of Natural products (Natural 5.1.1 for Windows, Natural 3.1.5 for Mainframes, Predict for Mainframes 4.2.1, etc.) in the conventional, platform-specific way. So you can profit from changes/enhancements made to these versions.

You can decide to change over to the Natural SPoD scenario at a later point of time. All you need to do is to install the remote development server plug-in (NDV) for the target platform and the desired plug-in(s) for Natural Studio, for example the XRef GUI Client.

There will be certain features in Natural Studio (for example, the application view) which you do not need in a single platform Natural development environment. In case of doubt, read the help texts. SPoD-specific functionality will be marked accordingly.

What is the difference between the SPoD application concept and the application shell concept existing in Natural for Windows?

The application shell is used with the frame gallery for developing, administering, and monitoring application solutions. Together, they provide an infrastructure in which numerous standards are defined and are implemented in the form of reuseable components, such as dialogs and procedures.

The application concept introduced with Natural Single Point of Development is intended to address the structure of customer solutions so as to get a new **logical** view on the solution which is not limited by the boundaries of the Natural libraries.

Natural Development Server - Overview

The documentation in the list below applies to the Natural Development Server for mainframes in general and to the Natural Development Server Plug-In under OS/390 in particular. Development servers for other platforms and operating systems are planned.

Natural Development Server Documentation

- What's New with Version 1.1.2 ?
- Introducing the Natural Development Server
- Development Server File
- Natural Development Server on Mainframes
- Development Server Installation under OS/390 *
- Development Server Configuration under OS/390 *
- Operating the Development Server under OS/390 *
- NDV Server Frequently Asked Questions (under OS/390)*

* Servers for other platforms and operating systems are under development.

What's New with Version 1.1.3 ?

This document describes the enhancements and corrections that apply to the release of Version 1.1.3 of the Natural Development Server.

The following topics are covered:

- Product Enhancements and Corrections
- Documentation

See also What's New with **Version 1.1.2** in the RN Archive on the Natural Documentation CD.

Product Enhancements and Corrections

This version contains

- all ZAPs,
- INPL updates,
- early warnings and
- source changes

applied to the predecessor version as error corrections.

Documentation

The installation instructions have been revised.

Introducing the Natural Development Server

This document describes the purpose and the functions of the Natural Development Server.

The following topics are covered:

- Purpose of a Development Server
- Remote Development Functions

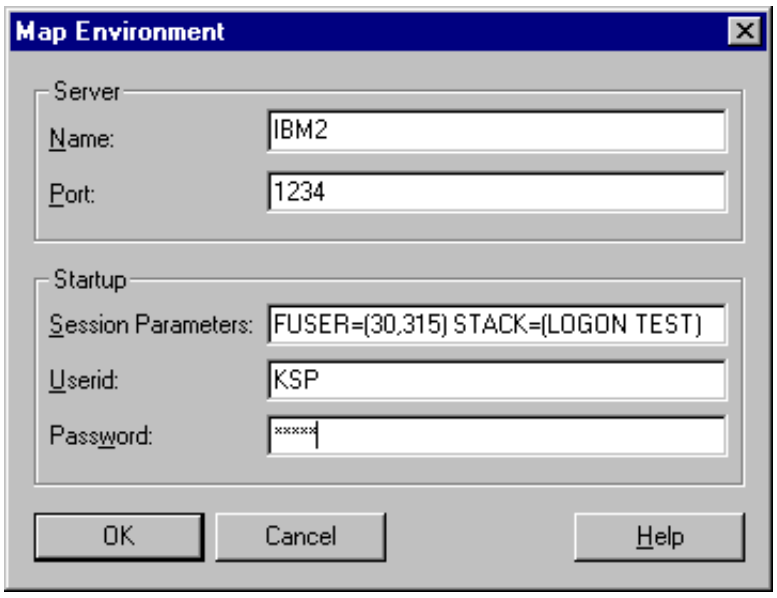
Purpose of a Development Server

With the Natural Development Server, you can use the Natural Studio development environment to develop and test Natural applications in a remote Natural environment.

Remote Development Functions

In the**Tools** menu, Natural Studio offers you a function named **Map Environment**. This function enables you to open a Natural session on a remote development server.

When the **Map Environment** function is invoked, a selection box appears where you can specify the required parameter to connect an active Development Server.

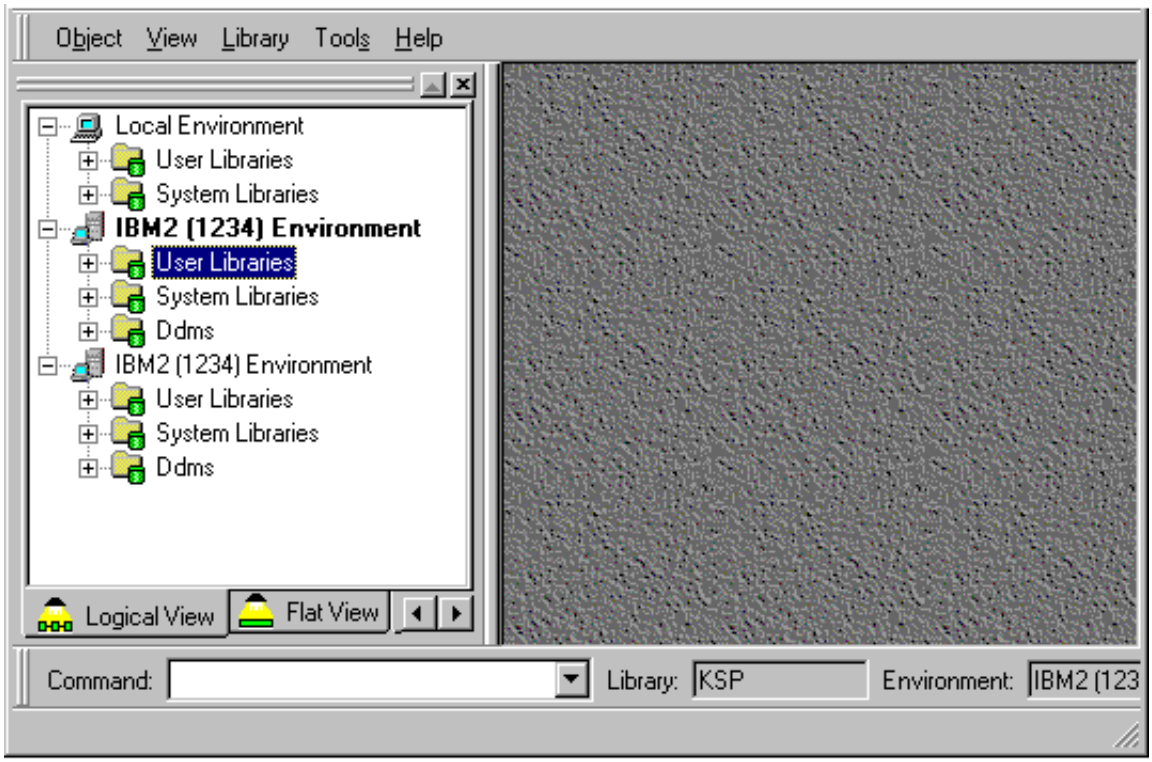


The server name defines the remote node name where the server is running.

The port defines the TCP/IP port number for the server. The port number is a configuration parameter of the Development Server.

You can specify additional Natural session parameters and a Natural Security password. Both are transferred to the Development Server and evaluated to create an exclusive Natural session that is responsible for executing all development requests for that environment.

Once you have executed a **Map Environment**, Natural Studio opens a new tree view that represents the Natural objects of the connected Natural development environment, for example:



Note: The node of a mapped server will reappear in the tree view when a new Natural session is started, unless it is deleted with the corressponding context menu function.

You can use the entire functionality of Natural Studio to create, edit, stow or execute Natural objects on the remote Natural environment.

You can map to multiple environments from one active Natural Studio. Each mapped environment owns a Natural session on the Development server, even if you map multiple environments on the same server. However, only one mapped environment can be active at a time. The active environment is shown in the status bar. In addition, the corresponding node is highlighted in the tree view. It is this environment in which all remote cammands will be executed.

You can change the active environment, by simply clicking on the desired node in the tree view.

In the example above, the user has mapped two environments on two different servers. Each environment owns a Natural session on the server, probably with different session parameters.

Development Server File

This document describes the purpose and use of the Development Server File, a central dictionary file which is used to store applications and the links to objects making up an application. It also holds object locking information in a Single Point of Development environment.

The following topics are covered:

- Purpose of the Development Server File
- Relations between FDIC and the Development Server File
- Unique Development Server File

Purpose of the Development Server File

As Natural stores its data in system files, the Natural Development Server stores its data in the system file that is assigned to the Natural parameter FDIC, a logical system file which is called the Development Server File.

The Development Server File is used as a central dictionary file for storing Natural applications and the links to objects making up an application. It also holds object locking information. This information is not bound to certain groups of application developers, but has an impact on the entire application development of an enterprise. Therefore, this file should be available only once, to ensure that the application definitions and locking states are kept consistent.

Relations between FDIC and the Development Server File

The Development Server File layout corresponds to the file layout of the Natural system file FDIC used as of Predict Version 4.2. This means that the central dictionary file can be also used to hold Predict data, but Predict is not a prerequisite for using the Development Server File. This enables you to use your existing application documentation in the application definitions of the remote development environment.

Unique Development Server File

It is of vital importance that the various development environments use a common and unique development server file. Non-compliance with this requirement may give rise to inconsistencies in object locking and in the applications existing in the application workspace.

To prevent the FDIC parameter from being overwritten when a development server is mapped, you are strongly recommended to disallow the specification of FDIC as a dynamic parameter using the NTDYNP macro.

In a development server which is protected by Natural Security, however, the use of another FDIC in the application workspace is prevented if the application security profiles are activated. See also Application Protection (in the Natural Security documentation).

Natural Development Server on Mainframes

This document describes the concept and the architecture of the Natural Development Server which is designed for use under the operating system OS/390.

The following topics are covered:

- Development Server Concept
- Front-End Stub NATRDEVS
- Transaction Processors
- Front-End
- Gateway Module
- Server Monitor

Development Server Concept

The Natural Development Server is a multi-user, multi-tasking application. It can host Natural sessions for multiple users and execute their requests concurrently.

The concept is based on the "serverized" Natural runtime system. It follows the architecture to have a server front-end stub (development server stub NATRDEVS) which uses the Natural front-end to dispatch Natural sessions and to execute functionality within these sessions.

The remote development server architecture basically consists of:

- Front-end stub
  The stub NATRDEVS is launched to initialize a development server, listens for incoming transactions and dispatches the appropriate Natural session for executing the transaction.
- Front-end
  It is called (together with the Natural runtime) by the front-end stub for session initialization/termination, request execution and session rollin/rollout.
- Gateway module
  The module NATGWSTG provides for interaction between the Natural runtime and the front-end stub. NATGWSTG is linked to the Natural nucleus and is called by the Natural runtime to exchange the necessary request data.
- Transaction processors
  are called by the front-end stub. The application logic of each individual transaction is implemented within a transaction processor.
- Server monitor
  A monitor task allows the administrator to control the server activities, to cancel particular users or to terminate the entire server, etc.

Front-End Stub NATRDEVS

The multi-user multi-tasking front-end stub NATRDEVS is launched to initialize a Natural Development Server.

- Stub Description
- Natural System Variables Used
- Natural I/O Handling

Stub Description

The task executing the server initialization (TMain) basically is the **main listener** who waits for incoming requests from the Remote Development Client. It owns a session directory to manage the multiple clients and their corresponding Natural sessions. TMain has the task to accept all incoming requests and to dispatch them to other subtasks (TWork). The process is as follows:

- A **map environment** command issued on the client side first connects to TMain to establish a connection.
- TMain inserts the client into its session directory, attaches a new TWork and passes the connection to TWork.
- TWork processes the request (indeed initializes a new Natural session if the client sends a CONNECT request) and replies to the client.
- After the reply, TWork listens on that connection for successive requests of that particular client.

That is, each client owns a subtask TWork on the Natural Development Server. This subtask remains active as long as the mapped environment on Natural Studio is the **current active environment**.

Natural System Variables Used

Within the Natural Development Server session, the following Natural system variables are used:

- *TPSYS contains ’SERVSTUB’,
- *DEVICE contains ’VIDEO’ and
- *SERVER-TYPE contains ’DEVELOP’.

Natural I/O Handling

The Natural runtime allows I/O execution in the same way as in an online environment:

- The Natural Development Server intercepts the I/O and sends the 3270 data stream to Natural Studio.
- Natural Studio internally starts a terminal emulation window and passes the 3270 stream to that window.
- After I/O execution, the I/O data is sent back to the server.
- The front-end stub invokes the front-end to continue processing after I/O.

Front-End

The Natural front-end required for a Natural Development Server is a Natural batch driver assembled with the option LE370=YES.

Transaction Processors

The Transaction Processors are Natural programs on library SYSLIB that are processing the transactions (e.g. "save source", "get library list",) demanded by the Development Server client. The Transaction Processors are invoked by the Front-End Stub.

Gateway Module

The gateway module NATGWSTG must be linked to the Natural nucleus.

Server Monitor

To enable the administrator to monitor the status of the Natural Development Server, a monitor task is provided which is initialized automatically at server startup. Using the monitor commands, the administrator can control the server activities, cancel particular users, terminate the entire server, etc. See Operating the Development Server.

Natural Development Server Installation under OS/390

This document describes how to install a Natural Development Server under the operating system OS/390.

The following topics are covered:

- Prerequisites
- Content of the Development Server Distribution Tape
- Installation Procedure

Prerequisites

- OS/390 must be installed.
  Version as specified under Operating/Teleprocessing Systems Required in the current Natural Release Notes.
- The current Natural version for Mainframes must be installed.
- If you are using Predict and you have to migrate to a Predict version as specified under Natural and Other Software AG Products in the current Natural Release Notes, you are strongly recommended to migrate to the newer Predict version **before** you install the Natural Development Server.
- List XRef Version 1.1.1
  (List XRef is an internal product with product code PXR and is included in the delivery for the Natural Development Server)
- The Software AG Editor must be installed. You are recommended to set the size of the editor buffer pool to 1024k.
  If you are using SMA, the necessary modules are linked when the SMA parameter SAG-EDITOR is set. If you are installing without SMA, see the Natural Installation Guide for Mainframes, Installing the Software AG Editor.

If you do not migrate to a Predict version as specified under Natural and Other Software AG Products in the current Natural Release Notes **before** starting the Natural Development Server installation, you will have to define a new Natural system file (FNAT) and a new development server file (FDIC). The current Natural version for Mainframes and the desired additional products must have been loaded on the Natural system file FNAT before you start the installation of the Natural development server.

The prerequisites required for the operation of a Remote Development Client must be fulfilled in addition. For details, refer to the Natural for Windows Version 5.1 documentation.

Content of the Development Server Distribution Tape

The installation tape contains the datasets listed in the table below. The sequence of the datasets and the number of library blocks needed are shown in the **Report of Tape Creation** which accompanies the installation tape.

There are two components on the tape:

- Natural Development Server
- System Management Hub for Natural

The following datasets refer to the Natural Development Server. For details on the System Management Hub, refer to System Management Hub for Natural Manager.

dataset Name	Contents
NDV*nnn*.LOAD	Contains the load modules of the development server. See Natural Development Server on Mainframes.
NDV*nnn*.EXPL	Contains the sample programs required for using the tutorial. See First Steps with Natural Single Point of Development.
NDV*nnn*.INPL	Contains the transaction processor. See Natural Development Server on Mainframes.
NDV*nnn*.ERRN	Contains the error messages of the transaction processor.
NDV*nnn*.SYSF	Contains the FDT of the Development Server File (the layout is identical with NDV*nnn*.SYSF provided with a Predict version as specified under Natural and Other Software AG Products in the current Natural Release Notes).

The notation *nnn* in dataset names represents the version number of the product.

Installation Procedure

To install the Natural Development Server in the OS/390 environment, perform the following steps:

Copying the Tape Contents to Disk

If you are using System Maintenance Aid (SMA), refer to the SMA documentation (included on the current edition of the Natural documentation CD).

If you are **not** using SMA, follow the instructions below.

This section explains how to:

- Copy data set COPY.JOB from tape to disk.
- Modify this data set to conform with your local naming conventions.

The JCL in this data set is then used to copy all data sets from tape to disk.

If the datasets for more than one product are delivered on the tape, the dataset COPY.JOB contains the JCL to unload the datasets for all delivered products from the tape to your disk.

After that, you will have to perform the individual install procedure for each component.

Step 1 - Copy data set COPY.JOB from tape to disk

The data set COPY.JOB (label 2) contains the JCL to unload all other existing data sets from tape to disk. To unload COPY.JOB, use the following sample JCL:


```
//SAGTAPE JOB SAG,CLASS=1,MSGCLASS=X
//* --------------------------------
//COPY EXEC PGM=IEBGENER
//SYSUT1 DD DSN=COPY.JOB,
// DISP=(OLD,PASS),
// UNIT=(CASS,,DEFER),
// VOL=(,RETAIN,SER=<Tnnnnn>),
// LABEL=(2,SL)
//SYSUT2 DD DSN=<hilev>.COPY.JOB,
// DISP=(NEW,CATLG,DELETE),
// UNIT=3390,VOL=SER=<vvvvvv>,
// SPACE=(TRK,(1,1),RLSE),
// DCB=*.SYSUT1
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//
```

Where:

<*hilev*>	is a valid high level qualifier
<*Tnnnnn*>	is the tape number
<*vvvvvv*>	is the desired volser

Step 2 - Modify COPY.JOB to conform with your local naming conventions

There are three parameters you have to set before you can submit this job:

- Set HILEV to a valid high level qualifier.
- Set LOCATION to a storage location.
- Set EXPDT to a valid expiration date.

Step 3 - Submit COPY.JOB

Submit COPY.JOB to unload all other data sets from the tape to your disk.

Note that the copy job copies the datasets of both the Natural Development Server and the System Management Hub for Natural.

Step 1: Apply the following corrections before you start to install NDV

1. If you have Predict Version 4.2.1 installed, apply the PRD421 Summary IUPD Update (PD421In) to your Natural system file (FNAT).
2. If you have PXR 1.1.1 installed, apply the PXR Summary IUPD Update (PX111In) to your Natural system file (FNAT).

Step 2: Create a development server configuration file and sample Clist

(Job I009 / Step 8410,8420,8430)

See sample member NDVCONFG on dataset NDV*nnn*.JOBS.

Step 8430 creates a sample batch job to PING and TERMINATE an NDV server. See sample member NDVBATCH on dataset NDV*nnn*.JOBS.

Described in the section Development Server Configuration under OS/390.

The following parameters of the configuration file have to be defined. For the other parameters, the default values may be used:

FRONTEND_NAME	Specify the name of the NDV server frontend module you generate in Step 6.
PORT_NUMBER	Specify the TCP/IP port number under which the server can be connected..

Step 8420 creates a sample CLIST to PING and TERMINATE an NDV server. See sample member NDVCLIST on dataset NDV*nnn*.JOBS.

Step 3: Load FDIC system file

(Job I050, Step 8403)

If you do not use Predict at all or if you have not yet migrated to a Predict version as specified under [MF::MF]Natural and Other Software AG Products[MF::MF] in the current Natural Release Notes, create the development server file, using the dataset NDV*nnn*.SYSF.

The layout of the Development Server File corresponds to the layout of the Predict Version 4.2 or above dictionary file.

Note: If you have a Predict version installed as specified under [MF::MF]Natural and Other Software AG Products[MF::MF] in the current Natural Release Notes, you can ignore this step.

Step 4: Assemble and link ADALNK

(Job I055, Step 8401)

The server environment requires a reentrant ADALINK module.

Link ADALNK using the RENT option.

Step 5: Assemble NATOS with LE370=YES

(Job I055, Steps 8410, 8420)

- Job I055, Step 8410 starts the batch program IEBUPDATE to create the source member.
- Job I055, Step 8420 assembles and links NATOS.
- If Predict XRef data is to be used:
  set the XREF parameter to ON or FORCE.

Adapt your Natural parameter module with the new parameters and assemble it.

Link the new Natural parameter module and, if Predict is to be used, the PRDXREF module from the load library NAT31n.LOAD to the environment-independent part of the Natural nucleus.

Step 6: Create NATPARM and NDV server frontend module

(Job I060, Steps 8410, 8420, 8430)

- Job I060, Step 8410 starts the batch program IEBUPDATE.
- Job I060, Step 8420 assembles and links NATPARM.
- Job I060, Step 8430 link the NDV server frontend module.

Step 7: Load Natural objects, error messages and samples for NDV

(Job I061, Steps 8450,8451,8450)

During NDV INPL the assigned FDIC/FSEC file is initialized with NDV specific information.

- Load objects from dataset NDVnnn.INPL onto your Natural system file (FNAT) using the INPL command. The parameter FDIC must have been set to point to your development server file.
- Load the error messages from dataset NDV*nnn*.ERRN using ERRLODUS.
- To use the tutorial (see First Steps with Natural Single Point of Development), load the sample programs from dataset NDVnnn.EXPL to your Natural system file.

Step 8: Load Natural objects for Predict XREF

(Job I061, Step 0601)

If you do not have PXR Version 1.1.1 installed yet, perform this step.

Load PXRnnn.INPL.

Note: If you are using Predict, but do not have a Predict version installed as specified under [MF::MF]Natural and Other Software AG Products[MF::MF] in the current Natural Release Notes and you intend to use existing XRef data, you have to recatalog the applications which will use the development server file.

Step 9: Copy DDMs and processing rules to FDIC

If you use a Predict Version 4.2 system file FDIC as development server file (FDIC), ignore this step.

If a Predict version as specified under [MF::MF]Natural and Other Software AG Products[MF::MF] in the current Natural Release Notes has not been installed or if you do not use a Predict Version 4.2 system file FDIC as development server file (FDIC), you have to copy the existing DDMs and processing rules to the development server file (FDIC), using the copy function of the Natural utility SYSMAIN.

Step 10: Create server startup JCL

(Job I200, Step 8415)

Described in the section Development Server Configuration under OS/390. See sample member NDVSTART on dataset NDV*nnn*.JOBS.

Sample:

```
//         PROC SRV=SAGNDV
//NDV      EXEC PGM=NATRDEVS,
// REGION=4000K,TIME=1440,PARM='POSIX(ON),TRAP(ON,NOSPIE)/&SRVŸ
//STEPLIB  DD   DISP=SHR,DSN=NDVvrs.LOAD
//         DD   DISP=SHR,DSN=SMA.LOAD
//SYSUDUMP DD   SYSOUT=X
//CEEDUMP  DD   SYSOUT=X
//CMPRINT  DD   SYSOUT=X
//STGCONFG DD   DISP=SHR,
//              DSN=NDV.CONFIG(&SRV)
//STGTRACE DD   SYSOUT=X
//STGSTDO  DD   SYSOUT=X
//STGSTDE  DD   SYSOUT=X
//SYSOUT   DD   SYSOUT=X
```


Note: The NDV server account must be defined in OS/390 UNIX System Services (OE segment). If the server account is not defined, the server ends with U4093 and system message CEE5101C in the trace file.

Step 11: NDV Clients must be defined to Natural Security

If Natural Security (NSC) is installed and an NDV client is not defined, the map environment returns an NSC error.

If you logon to the server from an NDV client, make sure that the user who is defined in Natural Security has a default libary or a private library defined. Otherwise, error message NAT0815 will occur.

Development Server Configuration under OS/390

This document describes how to configure the Development Server.

The following topics are covered:

- Configuration File
- Configuration Parameters
- Server Datasets

Configuration File

A configuration file is allocated to the DD-name STGCONFG. The configuration file is a text file located on a dataset or an HFS file that contains the server configuration parameters in form of a *keyword=value* syntax.

Configuration Parameters

The following configuration parameters are available:

FRONTEND_OPTIONS | FRONTEND_PARAMETER | THREAD_NUMBER | FRONTEND_NAME | THREAD_SIZE | TRACE_LEVEL | SESSION_PARAMETER | DEFAULT_PROFILE | HOST_NAME | PORT_NUMBER

FRONTEND_OPTIONS

This configuration parameter specifies additional options for the Natural front-end as follows:

01	Do not use the roll server.
02	Clean up roll file at server termination.
04	Write GTF trace.
08	Write ETRACE.
10	Front-End automatic termination.
20	Write console information.

Default value: 01

Example:

```
FRONTEND_OPTIONS=07 switches on options 01, 02 and 04.
```

FRONTEND_PARAMETER

This optional configuration parameter specifies additional Natural Front-End parameters as specified in the Startup Parameter Area. You can define multiple parameters. Each parameter is specified by a pair of 8-character strings, of which the first contains the parameter keyword and the second the parameter value. For further information, see the Natural Operations for Mainframe documentation, Natural in Batch Mode.

Default value: none

Example:

```
FRONTEND_PARAMETER ="MSGCLASSX"
```

This setting determines that the default output class for CMPRINT is "X".

THREAD_NUMBER

This configuration parameter specifies the number of physical storage threads to be allocated by the Natural front-end, that is, the number of sessions which can be executed in parallel.

Note:
This number does not limit the number of sessions within the server, but the number of sessions which can be in execution status concurrently. The number of sessions is limited by the size of the Natural swap medium.

Default value: 3

Example:

```
THREAD_NUMBER=5
```

FRONTEND_NAME

This configuration parameter specifies the name of the Natural front-end to start a Natural session. The front-end resides on a PDS member.

Default value: none

Example:

```
FRONTEND_NAME=NAT315SV
```

THREAD_SIZE

This configuration parameter specifies the size (in KB) of each physical storage thread which contains the Natural session data at execution time.

Default value: 500

Example:

```
THREAD_SIZE=800
```

TRACE_LEVEL

See Trace Level details in the section Natural Development Server on Mainframe.

Default value: 0

Example:

```
TRACE_LEVEL=0x00000011
```


This setting switches on Bits 31 and 27.

DEFAULT_PROFILE

This optional configuration parameter defines a default profile. Specifying a parameter string in the **Map Environment** window of Natural Studio overwrites this default profile.

Default value: none

Example:

```
DEFAULT_PROFILE = RDEVS,10,930
```

If no parameters are defined in the **Map Environment** window, the session is started with the parameter PROFILE=(RDEVS10,930).

SESSION_PARAMETER

This optional configuration parameter defines session parameters that precede the parameter string either specified in the **Map Environment** window or defined by default from DEFAULT_PROFILE.

Default value: none

Example:

```
SESSION_PARAMETER = FNAT=(10,930)
```

Every session on this development server is started with the session parameter FNAT=(10,930) appended by the user-specified parameters or the definitions in DEFAULT_PROFILE.

HOST_NAME

This optional configuration parameter is necessary only if the server host supports multiple TCP/IP stacks.

If HOST_NAME is specified, the server listens on the particular stack specified by HOST_NAME, otherwise the server listens on all stacks.

Default value: none

Example:

```
HOST_NAME = node1
```

or

```
HOST_NAME = 157.189.160.55
```

PORT_NUMBER

This configuration parameter defines the TCP/IP port number under which the server can be connected.

Default value: none

Example:

```
PORT_NUMBER = 3140
```


Configuration file example:

```
# This is a comment
SESSION_PARAMETER= profile=(stgqa,10,930) fuser=(10,32)
DEFAULT_PROFILE = DEFPROF
THREAD_NUMBER = 2
THREAD_SIZE=700
FRONTEND_NAME  =   NATOS31L       # and another comment
PORT_NUMBER=4711
```

Server Datasets

The Development Server requires the following datasets:

STGCONFG	Defines the server configuration file.
STGTRACE	The server trace output.
STGSTDO	The stdo dataset.
STGSTDE	The stde error output.

Operating the Development Server under OS/390

This document describes how to operate a Natural Development Server in an OS/390-based mainframe environment.

The following topics are covered:

- Starting the Development Server
- Terminating the Development Server
- Monitoring the Development Server
- Runtime Trace Facility
- Trace Filter

Starting the Development Server

The development server can be started as a **started task**:

```
//NDVSRV  PROC
//KSPSRV   EXEC PGM=NATRDEVS,REGION=4000K,TIME=1440,
// PARM=('POSIX(ON)/NDVSRV1')
//STEPLIB  DD DISP=SHR,DSN=NDVvrn.LOAD
//         DD DISP=SHR,DSN=NATvrn.LOAD
//CMPRINT  DD SYSOUT=X
//STGCONFG DD DISP=SHR,DSN=NDV111.CONFIG(SRV1)
//STGTRACE DD SYSOUT=X
//STGSTDO  DD SYSOUT=X
//STGSTDE  DD SYSOUT=X
```

Where *vrn* is the version, release, system maintenance level number of NDV (starting with NDV111) or Natural (starting with NAT315).

Note: PARM=('POSIX(ON)/NDVSRV1')
POSIX(ON) is required for a proper LE370 initialization and NDVSRV1 is the name of the server for the communication with the System Management Hub.

The name of the started task must be defined under RACF and the OS/390 Unix System Services.

Terminating the Development Server

The Development Server can be terminated from within the Monitor Client NATMOPI.

Monitoring the Development Server

To enable the administrator to monitor the status of the Natural Development Server, a monitor task is provided which is initialized automatically at server startup. Using the monitor commands described below, the administrator can control the server activities, cancel particular users, terminate the entire server, etc.

Monitor Communication

To communicate with the monitor, you can use the Monitor Client NATMOPI (documented in the Natural for Mainframes Operations documentation).

Monitor Commands

The Natural Development Server supports the following monitor commands:

Monitor Command	Action
ping	Verifies whether the server is active. The server responds and sends the string "I'm still up".
terminate	Terminates the server.
abort	Terminates the server immediately without releasing any resources.
set configvariable value	With the **set** command, you can modify server configuration settings. For example, to modify SERVER_TRACE: `set SERVER_TRACE 0x00000012`
list sessions	Returns a list of active Natural sessions within the server. For each session, the server returns information about the user who owns the session, the session initialization time, last activity time and an internal session identifier (session-id).
cancel session session-id	Cancels a particular Natural session within the NDV server. To obtain the session-id, use the monitor command **list sessions**.
help	Returns help information about the monitor commands supported.

Runtime Trace Facility

For debugging purposes, the server code has a built-in trace which can be switched on if desired.

The following topics are covered:

- Trace Medium
- Trace Configuration
- Trace Level

Trace Medium

A remote development server writes its runtime trace to a dataset allocated to the DD name STGTRACE.

The trace file is allocated (overwritten) at server initialization.

Trace Configuration

The trace is configured by a trace level which defines the detail of the trace. Once a trace is switched on, it can be restricted to particular clients or client requests by specifying a trace filter.

Every Natural session is equipped with a 32-bit trace status word (TSW) which defines the trace level for that session. The value of the TSW is set by a server configuration parameter. A value of zero means that the trace is switched off.

Trace Level

Each bit of the TSW is responsible for certain trace information. Starting with the rightmost bit:

Bit 31	Trace main events (server initialization/termination, client request/result).
Bit 30	Detailed functions (session allocation, rollin/rollout calls, detailed request processing).
Bit 29	Dump internal storage areas.
Bit 28	Session directory access.
Bit 27	Dump request/reply buffer.
Bit 26-24	Free.
Bit 23	Request processing main events.
Bit 22	Request processing detailed functions.
Bit 21-16	Free.
Bit 15	Trace error situations only.
Bit 14	Apply trace filter definitions.
Bit 13-08	Free.
Bit 07-01	Free.
Bit 00	Reserved for trace-level extension.

Trace Filter

In order to reduce the volume of the server trace output, it is possible to restrict the trace by a logical filter.

- The filter can be set with the configuration parameter TRACE_FILTER.
- The filter may consist of multiple keyword=*filtervalue* assignments separated by spaces.

The filter keyword is:

Client	Filters the trace output by specific clients.

The following rules apply:

- If a keyword is defined multiple times, the values are cumulated.
- The value must be enclosed in braces and can be a list of filter values separated by spaces.
- The values are not case sensitive and asterisk notation is possible.

Example:

```
TRACE_FILTER="Client=(KSP P*)"
```

Each request of the userid KSP and each request of the userids prefixed by a P are traced.

NDV Server Frequently Asked Questions

This document contains a number of frequently asked questions concerning the Natural Development Server (NDV) under OS/390.

The following topics are covered:

NDV server starts and terminates immediately

At server initialization, the NDV server

- allocates central control blocks,
- opens the datasets STGTRACE, STGSTDO, STGSTDE, STGCONFG,
- obtains the configuration file,
- loads the Natural Front-End,
- initializes the first Natural session and
- launches the TCP/IP listener task

If one of these steps fails, the server cannot continue and terminates immediately.

Analyze the trace output (STGTRACE) or the error output (STGSTE) to find out the problem.

Which dataset should I analyze to get error information?

STGSTE	Contains only error output. Each record consists of 2-4 lines depending on whether it is a Natural error, a system error or an NDV stub error. Natural Error 1. **DayOfMonth Time TaskId UserId** 2. **TaskId NDV Error:** error classification 3. **NATURAL FrontEnd error** or **NATURAL runtime error** 4. Natural error text System Error 1. **DayOfMonth Time TaskId UserId** 2. **TaskId NDV Error:** error classification 3. **TaskId Sys Error:** System error text NDV stub error 1. **DayOfMonth Time TaskId UserId** 2. **TaskId NDV Error:** error classification
STGTRACE	Contains NDV trace information and error information. Each trace record contains **DayOfMonth Time TaskId** Trace information text The string **PrintError** in the Trace information text prefixes errors.
STGSTO	Content of the configuration file allocated to STGCONFG.
SYSOUT	Messages from LE/370 runtime system.

Trace output shows "Cannot load NATURAL Front End "

The Natural Front-End specified by the NDV configuration parameter FRONTEND_NAME was not found in the load library concatenation.

Trace output shows "Transport initialization failed" "EDC8115I Address already in use"

The TCP/IP port number specified by the NDV configuration parameter PORT_NUMBER is already in use by another process.

How do I get information about which process occupies a port number?

TSO command **NETSTAT (PO 4712)** displays connections of Port 4712. The first column of the list refers to the task that owns the port.
Or enter the OS/390 Unix System Services command **netstat -P4712**.

The task that occupies a port number is not active but the port is still occupied. How do I drop the stuck connections?

Enter TSO command NETSTAT (PO *nnnn*) to list connections for port nnnn.

Output of the NETSTAT (PO 4712) command:

```
EZZ2350I MVS TCP/IP NETSTAT CS V2R8       TCPIP NAME: DAEFTCP2        06:45:19
EZZ2585I User Id  Conn     Local Socket           Foreign Socket         State
EZZ2586I -------  ----     ------------           --------------         -----
EZZ2587I SAGNDV31 000031CC 157.189.160.55..4712   192.168.40.11..3152    Establsh
EZZ2587I SAGNDV31 000005E9 0.0.0.0..4712          0.0.0.0..0             Listen
EZZ2587I SAGNDV31 000031CD 157.189.160.55..4712   192.168.40.27..4250    Establsh
EZZ2587I SAGNDV31 000031D5 157.189.160.55..4712   157.189.164.133..2906  Establsh
EZZ2587I SAGNDV31 000031D8 157.189.160.55..4712   157.189.164.152..1099  Establsh
```


User Id	The job that uses port 4712.
Conn	Connection ID.
Foreign Socket	Connected clients.
State	Connection status.

If State contains FinWait, you need not drop that connection, because connections of that status do not prevent an NDV server from using that port.

To drop the connection, enter the MVS command
`VARY TCPIP,DAEFTCP2,DROP,000005E9`.

Where DAEFTCP2 must match your TCP/IP job name (TCPIP NAME: DAEFTCP2) in the first line of the NETSTAT output) and 000005E9 is the connection ID in the column Conn.

Trace output shows "Error at:Template runtime connect"

When a Natural development server initializes, it starts a Natural session using the session parameter defined by the NDV configuration parameter SESSION_PARAMETER. The profile definition of the NDV configuration parameter DEFAULT_PROFILE is appended.
If the initialization of the template session fails, the server terminates immediately. The original error can be found below the message "Error at:Template runtime connect".

NDV task abends with User Code 4093 and SYSOUT Message CEE5101C

The account of the NDV server is not defined in OS/390 Unix System Services. If you start the NDV server as a started task, the member name of the started task must be defined under OS/390 Unix System Services. If you start the NDV server as a batch job, the user that submits the job must be defined under OS/390 Unix System Services.

Required LE/370 runtime options

LE/370 runtime options that must be specified to operate an NDV server.

POSIX(ON)	Enables the NDV server to access the POSIX functionality of OS/390. If you start a NDV server with POSIX(OFF), it terminates immediately with a user abend U4093 and the system message EDC5167. IBM supplies the default OFF.
TRAP(ON,NOSPIE)	Defines the abend handling of the LE/370 environment. ON enables the Language Environment condition handler. NOSPIE specifies that Language Environment will handle program interrupts and abends via an ESTAE, that is the Natural abend handler receives control to handle program interrupts and abends. If you do not specify TRAP(ON,NOSPIE) the Natural abend handling does not work properly. IBM supplies the default (ON,SPIE).
TERMTHDACT(UADUMP)	Defines the the level of information that is produced in case of an abend. The option UADUMP generates a Language Environment CEEDUMP and system dump of the user address space. The CEEDUMP does not contain the Natural relevant storage areas. IBM supplies the default (TRACE).

Useful LE/370 runtime option?

LE/370 runtime options to monitor and tune NDV servers.

RPTOPTS(ON)	Prints LE/370 runtime option settings to SYSOUT after server termination.
HEAPPOOLS	The HEAPPOOLS run-time option is used to control an optional heap storage management algorithm, known as heap pools. Refer also to Language Environment for OS/390 & VM Programming Reference. The setting of this parameter depends on NDV functionality mostly used by NDV clients. A good value to start with is: HEAPP=(ON,40,3,80,7,224,7,528,3,1344,8,2048,8).
ALL31(ON)	Specify ALL31(ON) if your entire Natural environment runs in 31-bit mode to prevent LE/370 switching addressing mode.
STACK(64K,16K,ANY,FREE)	Specify the ANY option if your entire Natural environment runs in 31-bit mode. This enables LE/370 to allocate the storage for the STACK segment above the 16 MB line. The STACK segment above 16 MB increases the number of subtasks you can create within the NDV region. The initial and extend size (64 KB and 16 KB in the example) should be determined for your own environment by using the LE/370 storage report generated when you specify RPTSTG(ON).
HEAP(800K,64K,ANY,FREE,,)	Initial heap storage (see STACK option).
ANYHEAP(1300K,200K,ANY,FREE)	Library heap storage (see STACK option).
RPTSTG(ON)	Generates, after server termination, a report of the storage the server used. At the end of the report, it suggests cell sizes for the HEAPPOOLS option. This option decreases performance of the server. Use it only as an aid to find best settings for HEAPPOOLS definition.
ENVAR(TZ=)	The ENVAR option enables you to set UNIX environment variables. The only environment variable applicable for the NDV server is TZ (time zone). **Example:** ENVAR(TZ=CET-1DST) CET - 1 hour daylight saving time

---

How do I pass LE/370 runtime options?

1. With the PARM parameter specified in the EXEC card of the NDV startup job.
   The length of the options is limited by the maximum length of the PARM parameter.

```
//NDV EXEC PGM=NATRDEVS,
// PARM='RPTOPTS(ON)/server-id'
```

2. Assemble an LE/370 runtime option module CEEUOPT and link it to the NDV load module.

```
//KSPLNDV  JOB  KSP,CLASS=K,MSGCLASS=X
//*
//* RELINK NDV SERVER WITH LE/370 RUNTIME OPTIONS
//*
//*****************************************************************
//* STEP1: ASSEMBLE LE/370 RUNTIME OPTION MODULE
```



```
//*
//STEP1 EXEC PGM=ASMA90,PARM='DECK,NOOBJECT'
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT2   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPUNCH DD DSN=&&TEMPOBJ(CEEUOPT),DISP=(,PASS),UNIT=SYSDA,
// SPACE=(TRK,(1,1,1)),DCB=(BLKSIZE=3120,LRECL=80,DSORG=PO)
//SYSLIB   DD DSN=CEE.SCEEMAC,DISP=SHR                    <<<<<<
//         DD DSN=SYS1.MACLIB,DISP=SHR                    <<<<<<
//SYSIN    DD *
CEEUOPT  CSECT
CEEUOPT  AMODE ANY
CEEUOPT  RMODE ANY
         CEEXOPT ENVAR=(TZ=CET-1DST),                                  X
               HEAPPOOLS=(ON,40,50,80,90,224,80,528,50,1344,90,2048,   X
               90),                                                    X
               POSIX=(ON),                                             X
               RPTOPTS=(ON)
         END
/*
//*****************************************************************
//* STEP1: LINK RUNTIME OPTION MODULE
//*
//STEP2    EXEC  PGM=IEWL,
//         PARM='NCAL,RENT,LIST,XREF,LET,MAP,SIZE=(9999K,96K)'
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(5,5))
//SYSLMOD  DD  DSN=&&CEEOBJ(CEEUOPT),DISP=(,PASS),UNIT=SYSDA,
// SPACE=(TRK,(1,1,1))
//SYSLIB   DD  DSN=&&TEMPOBJ,DISP=(OLD,PASS)
//SYSLIN   DD  *
 INCLUDE SYSLIB(CEEUOPT)
  ENTRY  CEEUOPT
   ORDER CEEUOPT
 NAME CEEUOPT(R)
/*
//*****************************************************************
//* STEP3: RELINK NDV SERVER WITH RUNTIME OPTION MODULE
//*
//STEP3    EXEC PGM=IEWL,
//         PARM='RENT,XREF,LIST,LET,REUS,SIZE=(300K,64K),CASE=MIXED,
//              AMODE=31,RMODE=ANY'
//SYSUT1   DD  UNIT=(SYSDA),SPACE=(TRK,(10,4))
//SYSLMOD  DD  DISP=SHR,DSN=NATURAL.NDV.LOAD      <<<<<<
//SYSPRINT DD  SYSOUT=X
//NDVLOAD  DD  DISP=SHR,DSN=NATURAL.NDV.LOAD      <<<<<<
//CEELOAD  DD  DISP=SHR,DSN=&&CEEOBJ
//SYSLIN     DD *
 REPLACE CEEUOPT
 INCLUDE NDVLOAD(NATRDEVS)
 INCLUDE CEELOAD(CEEUOPT)
 NAME NATRDEVS(R)
/*
```

The lines marked with <<<<<< must be adapted to your environment.

Definitions required in Natural Security

- Each client must be defined in Natural Security if the Transition Period Logon flag in NSC is set to NO. Otherwise, your Map Environment fails with a NAT0873.
- Each user must have either a default library or a private library. Otherwise your Map Environment fails with a NAT1699.
- You must not specify a startup program that executes an I/O statement or stacks a LOGON, LOGOFF or RETURN command, because the program is executed whenever you change the focus to that library within the tree view.
- If you add a new user, you must specify a password for this user. Otherwise, the Map Environment fails with a NAT0838.

I do not get a NAT0954 even if I specify DU=OFF

The LE/370 runtime option TRAP must be set to TRAP(ON,NOSPIE).

Map Environment fails with a NAT3048

Specify session parameter ETID=' '. If you have Natural Security, clear the ETID definition for that user.

Map Environment fails with Stub RC nn

Stub return codes are raised by the NDV Front-End stub if it detects a logical processing error when dispatching the NDV request. The NDV trace output contains detailed information about the reason for the error.

The following Stub return codes are possible:

1	**Error during session reconnect** (for future use).
2	**Cannot create new session directory entry or subtask.** If the Natural Studio executes a Map Environment, the NDV server allocates an entry in its session directory and creates a new subtask. If one of these actions fails, the Stub RC 2 is raised. **Reason:** • Region size (virtual storage below 16 MB) for the NDV server is too small, • Number of subtasks exceeds the limit specified by the OS/390 Unix System Services parameter MAXTHREADS. **Action:** Increase region size or MAXTHREADS, or distribute the clients to several NDV server. To save memory below 16 MB, you can also specify the ANY option of the LE/370 parameter STACK (refer to Useful_LE_runtime options). The number of active tasks can be displayed using the OS/390 system command **D OMVS,PID=*process-id*** (where *process-id* is the process id of the NDV server). The value of MAXTHREADS can be displayed with **D OMVS,OPTIONS**.
3	**Cannot initialize new session.** This error occurs if a storage allocation for internal NDV control buffers fails due to a lack of virtual memory above 16 MB. **Reason:** Virtual memory above 16 MB too small. **Action:** Increase the virtual memory above 16 MB, decrease the number of physical storage threads, configure NDV to use the Natural roll server, or distribute the clients to several NDV servers.

4	**Session execution failed.** Internal error. The Natural Studio uses an invalid session identifier to process a request. **Reason:** ● On Map Environment the session ID already exists. ● The Natural session with specified ID is not initialized. **Action:** Locate the defective session ID in the server trace file and cancel it using the monitor, or restart your Natural Studio.
5	**I/O execution not allowed.** In some situations, a Natural IO is prohibited at the NDV server. **Reason:** ● IO execution during LOGON request, ● IO execution during execution of a transaction processor. **Action:** Locate the IO buffer in the server trace file to find out which IO should be processed. Check for any startup program specified for the library you want to logon.
6	Not applicable.
7	**Error during I/O execution.** The NDV server cannot finish a terminal IO. **Reason:** ● Virtual memory above 16 MB too small, ● IO reply buffer send by Natural Studio is invalid. **Action:** Increase the virtual memory above 16 MB. If the IO reply buffer is invalid, contact Software AG support.
8	**Protocol element missing.** Internal error, contact Software AG support.
9	**NDV not installed on Natural systemfile.** NDV server cannot execute the Natural module TRPRO located on library SYSLIB. **Reason:** The NDV modules are not loaded on the FNAT. **Action:** INPL the NDV modules.
10	**LOGON command required.** If you execute a program on the NDV server that executes a LOGOFF (or a RETURN when no SETUP record is available), the logon library is undefined. In an online environment the Natural Security logon screen is displayed in this situation. Under NDV the Natural session rejects all requests except a LOGON command. This applies only if Natural Security is installed. You can execute a LOGON command either by using the command line or by clicking on any library in your tree view.

Special characters are not translated correctly

The ASCII-EBCDIC translation for NDV uses the Natural translate tables TABA1/TABA2. These tables can be maintained at customer site. The translate tables can be modified as follows:

1. Modify source member NTTABA1/NTTABA2 on the Natural distribution library. Reassemble NATCONFG and relink the Natural nucleus.
2. Specify the Natural session parameter TABA1/TABA2.

How do I find out which hexadecimal value must be specified for TABA1/TABA2?

Run the following program on your Natural for Windows locally.

```
#A(A1) = '{'
   WRITE A(EM=H)
   END
```

Output is **7B**.

Run the program on a mainframe (edit the program with the Natural mainframe editor).
Output is **75**, assuming that you use a German EBCDIC table. If you use a US EBCDIC table, the output will be C0.

Start your NDV server session with TABA1=(75,7B) and TABA2=(7B,75).

First Steps with Natural Single Point of Development - Overview

First Steps with Natural Single Point of Development - Overview

This tutorial provides an introduction to Natural Studio which is part of the Single Point of Development (SPoD) concept. It is intended for mainframe Natural developers who are not yet familiar with the Windows version of Natural.

This tutorial is not intended to be a comprehensive description of the full range of possibilities provided by Natural Studio. Therefore, explanations are kept to a minimum. For detailed information, see Natural Studio in the Natural for Windows User's Guide.

It is recommended that you follow the tutorial in the sequence indicated below.

First Steps with Natural Single Point of Development

- Starting Natural
- Customizing the Natural Studio Window
- Local and Remote Environment
- Issuing Commands
- Handling Programs
- Locking and Unlocking
- Handling Applications
- Displaying Cross-Reference Data

Estimated duration for this tutorial: 2 hours.

Starting Natural

This tutorial assumes the following:

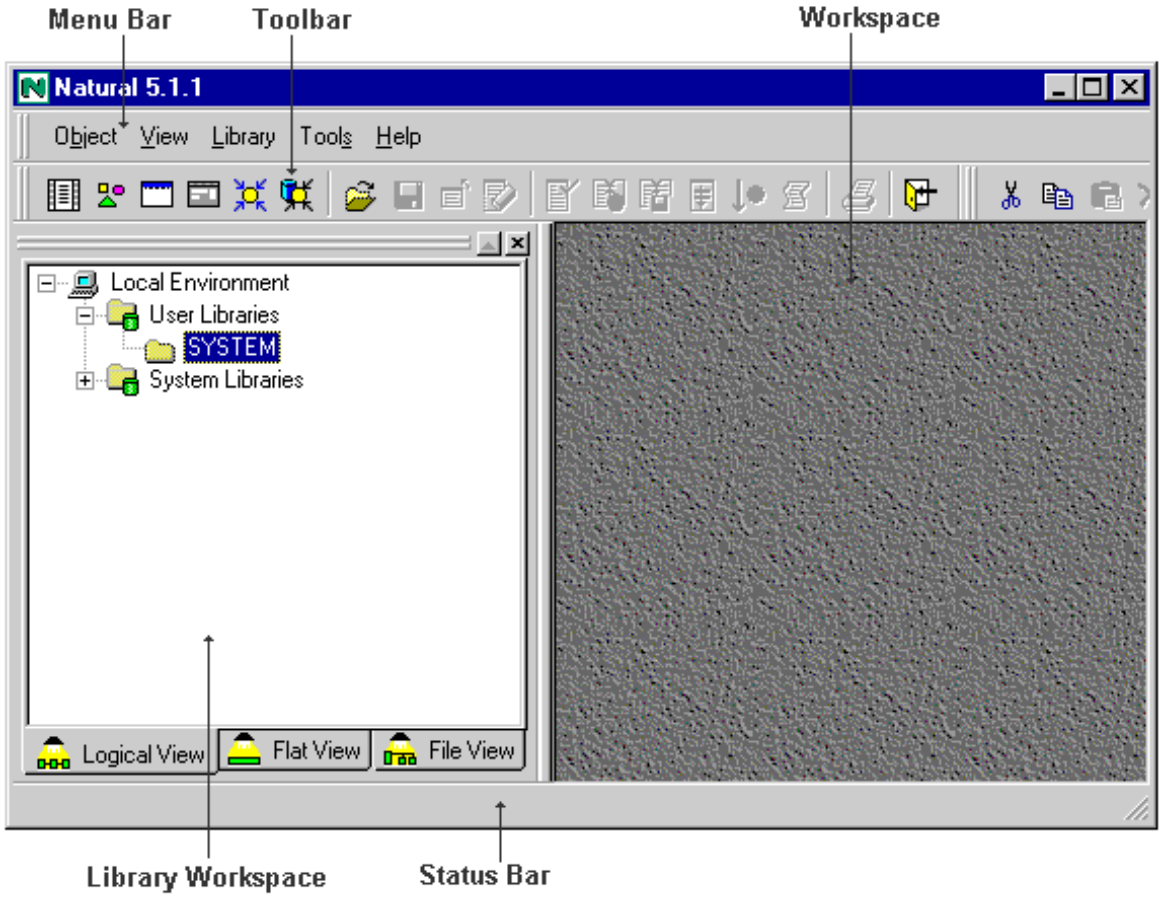
- Natural 5.1.1 for Windows has been installed with setup type "Development Client for Single Point of Development (SPoD)" .
- A development server has been installed on the mainframe. See Natural Development Server Installation under OS/390.
- The dataset NDV111.EXPL has been loaded with INPL on the development server. This dataset contains the sample libraries SYSSPODA and SYSSPODX that will be used in this tutorial.
- You have a basic understanding of how to use Microsoft Windows.
- You have already read Introducing Natural Single Point of Development.

To start Natural

- From the Start menu, choose **Programs** > **Software AG Natural 5.1.1** > **Natural**.
  Or double-click the following shortcut on your Windows desktop (only available if specified during installation).

  Natural Studio, the development environment for Natural, appears:



When you start Natural for the very first time, Natural Studio shows only your local environment containing the library workspace. The different environments are explained later in this tutorial.

You can now proceed with the first exercise: Customizing the Natural Studio Window.

Customizing the Natural Studio Window

This section provides the following information:

- Resizing Windows
- Moving and Docking Windows
- Using Different Views
- Displaying Additional Toolbars
- Displaying the Command Line

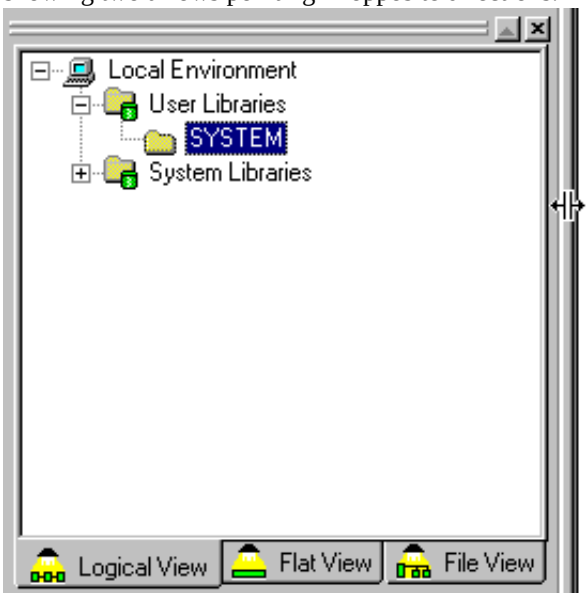
Resizing Windows

You can modify the size of each window within the Natural Studio window.

In this exercise, you will resize the library workspace window which is initially docked within the Natural Studio window.

To resize a docked window

1. Move the mouse pointer over the right border of the library workspace window until the pointer changes, showing two arrows pointing in opposite directions.

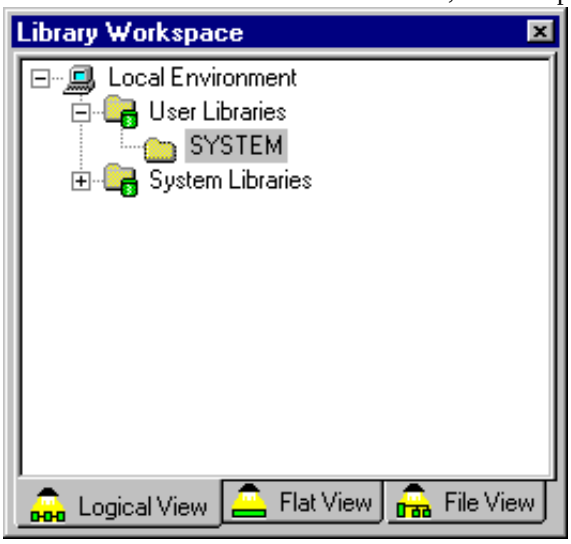


2. Click and hold down the mouse button.
3. Drag the mouse to make the window larger or smaller.
4. When the window has the desired size, release the mouse button.

Moving and Docking Windows

Using the mouse, you can drag each window within the Natural Studio window to another position:

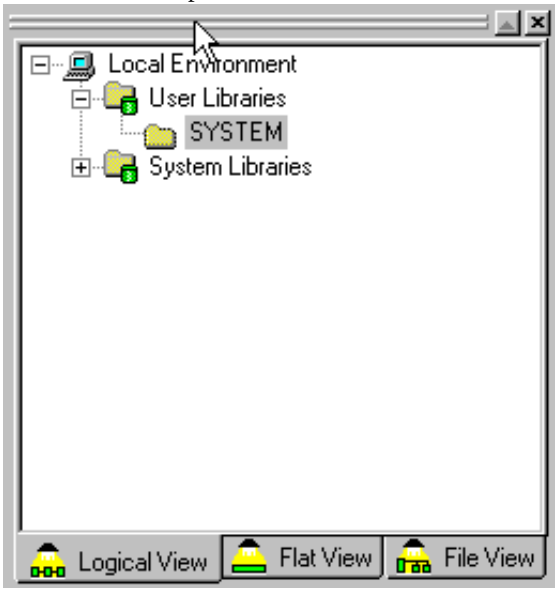
- so that it is shown at another position within the Natural Studio window, or
- so that it is shown in a window of its own, for example:



In the following exercises, you will undock, dock, hide and redisplay the library workspace window.

To move a docked window so that it is shown in a window of its own (undock)

1. Move the mouse pointer to the two horizontal lines at the top border of the library workspace window.



2. Click and hold down the mouse button.
3. Drag the mouse to move the window to a position outside the Natural Studio window.
   An outline of the window is shown.
4. Release the mouse button.
   The library workspace is now shown in a window of its own.

Note:
When you release the mouse button while the outline of the window is still shown within the Natural Studio window, it is docked at another position within the Natural Studio window. You can prevent docking by pressing CTRL while moving the mouse.

To move an undocked window back to the Natural Studio window (dock)

1. Click the title bar at the top of the undocked window and hold down the mouse button.
2. Drag the mouse to move the undocked window back to the Natural Studio window.
   An outline of the window is shown.
3. Release the mouse button.
   The window is now again docked within the Natural Studio window.

Note:
The position at which the window is docked depends on the position of the mouse pointer.

To hide the library workspace window

- Click the following button at the top right of the library workspace window (this can be either a docked or undocked window):
  
  The library workspace window is no longer shown in the Natural Studio window.

To toggle display of the library workspace window

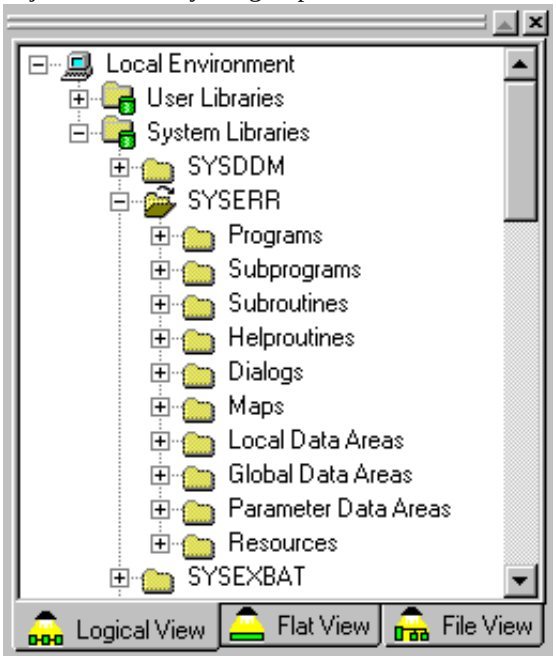
- From the **View** menu, choose **Library Workspace**.
  Or press ALT+1.
  When the library workspace window is displayed in the Natural Studio window, a check mark is now shown next to the **Library Workspace** command. The window is always restored in the same state it was before it was hidden (either docked or undocked).

Note:
When you exit Natural Studio, the settings in the **View** menu as well as position and size of the Natural Studio window and its subwindows are stored in the Windows registry. The next time you start Natural Studio, its window is restored as it appeared when you last used it.

Using Different Views

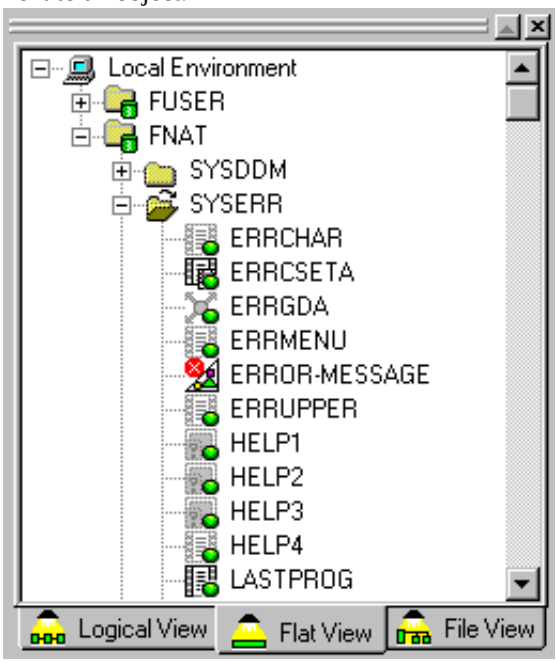
The library workspace window provides tabs for different views:

- **Logical View**
  In logical view, different nodes are provided for user libraries, system libraries and DDMs (FDIC). The objects in a library are grouped into different folders, according to their types.

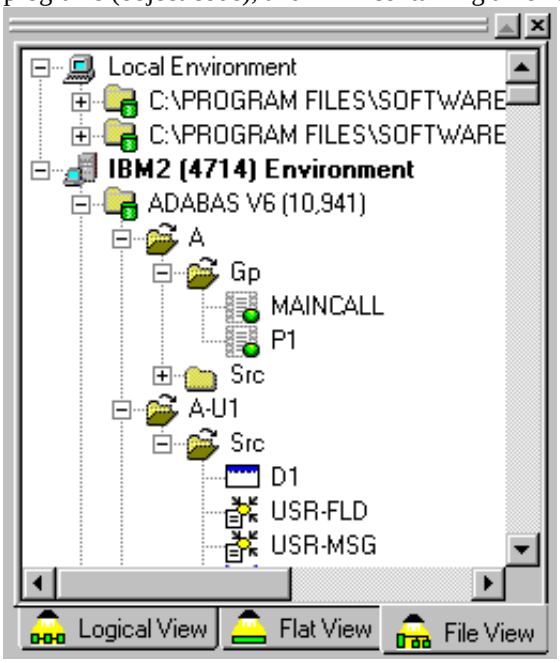


  For example, all programs are shown in a folder called "Programs". Thus, if you want to view the available programs in a library, you must first open the corresponding folder by clicking the plus sign next to the folder name. This is not required in flat view.

- **Flat View**
  In flat view, different nodes are provided for the Natural parameters: FUSER, FNAT and FDIC. The objects in a library are displayed without any grouping. However, the object type is always indicated by the icon next to an object.



- **File View**
  In file view, the name of each mapped remote database is shown with database ID and file number (mapping is explained later in this tutorial). Different folders are provided for each library (provided that the corresponding objects exist in this library): "Src" containing all Natural sources, "Gp" containing all generated programs (object code), and "Err" containing all error messages.



In the different views, icons are shown for the Natural objects. For example:

 This icon is shown for a program:

 When both source code and a generated program are available for an object, a green dot is shown on the icon.

 When only a generated program is available and no source code, the icon is gray.

Note:
If you want to see more icons for the different objects types, expand the node for SYSMAIN in logical view and then expand the nodes for the different folders.

When you resize the library workspace window, it may happen that the window is not wide enough to display all tabs at the bottom. In this case, arrow buttons are provided. To display another view, you can either enlarge the window and click the corresponding tab, or you can click an arrow button to display the corresponding view.

Displaying Additional Toolbars

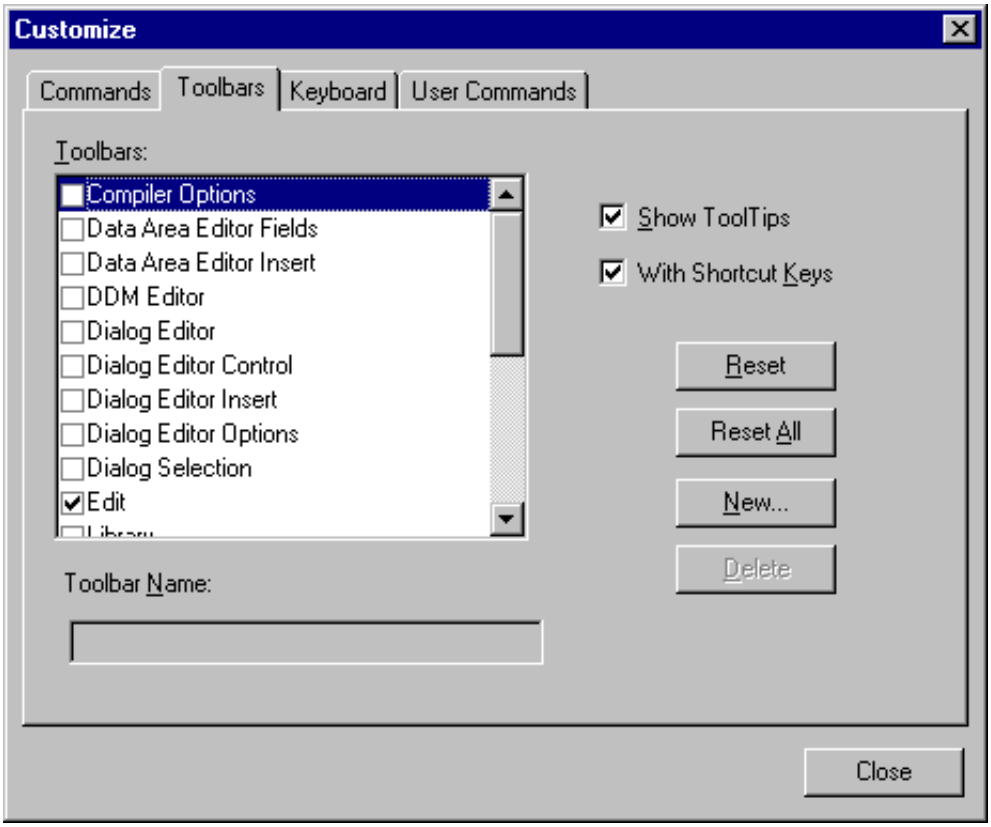
Natural Studio provides several toolbars. Only a few of them are initially shown in the Natural Studio window.

In this exercise, you will activate the Tools toolbar which provides buttons for commands that you will use later in this tutorial:

Note:
You can undock and dock each toolbar that is shown in the Natural Studio window as described above for the library workspace window.

To display the Tools toolbar

1. From the **Tools** menu, choose **Customize**.
   The "Customize" dialog box appears.
2. Click the "Toolbars" tab.



3. To proceed to the bottom of the list box, click the down arrow repeatedly.
   Or drag the scroll bar to the bottom of the scroll box.
   Initially, the "Tools" check box is not selected. This means that the Tools toolbar is currently not shown in the Natural Studio window.
4. Select the "Tools" check box so that a checkmark is shown in the box.
   Each toolbar that you activate is immediately shown in the Natural Studio window. It is always shown in its last state (either docked or undocked).
5. Choose the **Close** button.

Displaying the Command Line

You can issue all Natural commands directly from the command line. An example is given later in this tutorial.

Note:
You can undock and dock the command line as described above for the library workspace window.

Initially, the command line is not shown.

To toggle command line display

- From the **View** menu, choose **Command Line**.
  Or press ALT+3.
  When the command line is displayed in the Natural Studio window, a check mark is shown next to the **Command Line** command.

You can now proceed with the next exercise: Local and Remote Environment.

Local and Remote Environment

A Natural development environment contains all application components such as parameter modules, system files and buffer pool.

The following topics are covered below:

- Checking the Environment
- Connecting to a Development Server for the First Time
- Connecting to a Previously Mapped Development Server
- Logging on to a Library

Checking the Environment

SPoD supports the following types of environment:

- local environment on a workstation (this is also the runtime environment of Natural Studio)
- remote environment on a development server

You can check which of these two environments is currently active. The active environment is always indicated in the command line, next to the "Command" drop-down list box. When the command line is not shown, you can display it as described previously in Displaying the Command Line.

In the example below, the local environment is active and you are currently logged on the the library SYSTEM.

All commands that you issue are always applied to the active environment. When you edit a Natural object, the corresponding editor is invoked and the object is taken from the active environment. When you execute an object, it is executed in the active environment.

Only one environment can be active at one point in time.

Connecting to a Development Server for the First Time

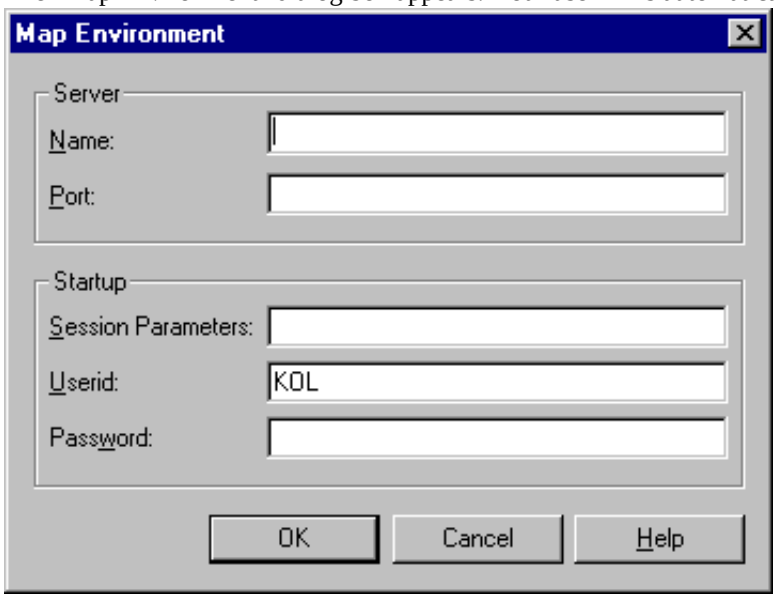
In order to perform remote development, you have to activate a remote Natural environment. You do this by connecting to the appropriate Natural development server. Each Natural development server provides all remote services (such as access or update) for a specific FUSER.

If you want to connect to a development server for the very first time, you have to map it as described below. Once you have connected to a development server, a node for this development server session is automatically shown in the tree the next time you invoke Natural Studio.

If you do not know the name and port number for your development server, ask your administrator before proceeding with the next exercise.

Note:
It is possible to map the same development server more than once, for example, if you want to have development server sessions with different session parameters. To switch to another session, you simply click the corresponding node in your library workspace.

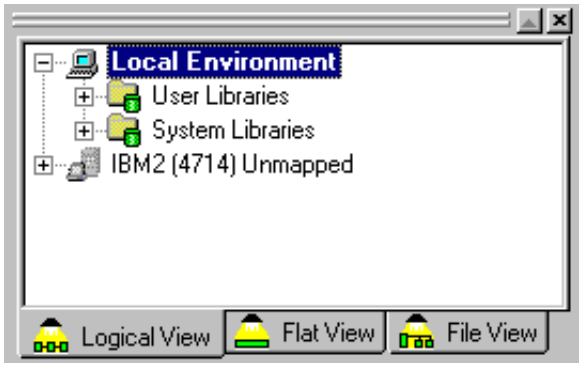
To connect (map) to a development server

1. From the **Tools** menu, choose **Map Environment**.
   Or click the following toolbar button.
   
   The "Map Environment" dialog box appears. Your user ID is automatically provided.
   

2. In the "Name" text box, enter the name of the development server on the mainframe.
3. In the "Port" text box, enter the TCP/IP port number of the development server.
4. If dynamic parameters are required for your development server, specify them in the "Session Parameters" text box. Otherwise, leave this text box blank.
5. If Natural Security is installed on the development server, specify the required password in the "Password" text box. Otherwise, leave this text box blank.
6. Choose the **OK** button.
   When the connection has been established, all libraries (according to the security profile) for this session are shown in your library workspace. You are automatically logged on to your default library. The command line now shows the name of the library that is currently selected in the tree and the name of the active environment (i.e. the name you specified for the development server on the mainframe).

Connecting to a Previously Mapped Development Server

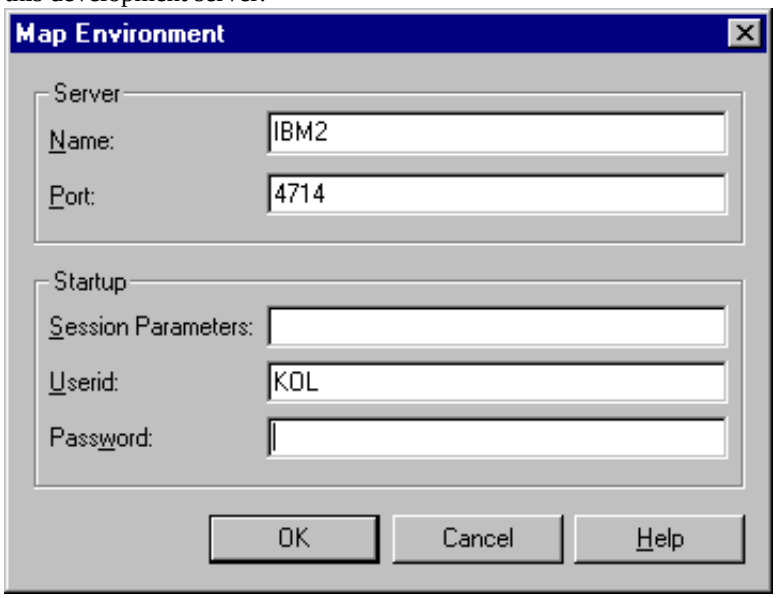
Once you have connected to a development server, its name is automatically shown as a node in the tree of your library workspace. Each time you restart Natural, the state of each development server is set to "unmapped". This information is shown in the tree.



Note:
It is also possible to delete an unmapped development server so that its name is no longer shown in the tree (select the server, click the right mouse button and from the resulting context menu, choose **Delete**).

To connect to a previously mapped development server

1. Exit Natural and start it once more.
   The development server you have previously mapped is now shown with the state "Unmapped".
2. Click the plus sign next to the node name.
   Or double-click the node name.
   The "Map Environment" dialog box appears. It shows the information that you have previously provided for this development server.



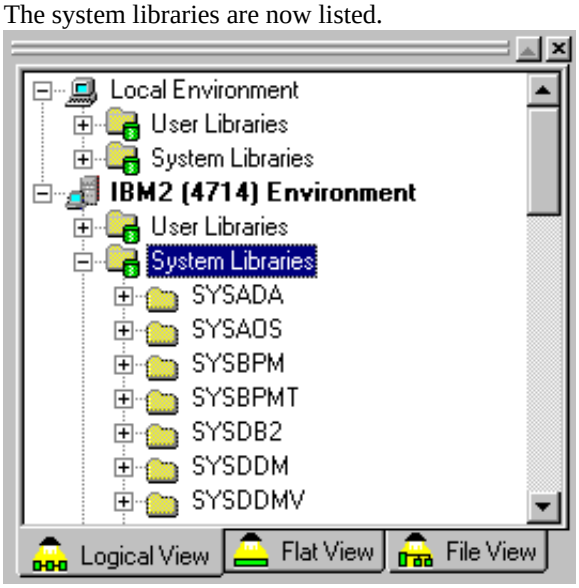
3. If Natural Security is installed on the development server, specify the required password in the "Password" text box. Otherwise, leave this text box blank.
4. Choose the **OK** button.
   The libraries for this development server session are now shown in the library workspace. You are automatically logged on to your default library.

Logging on to a Library

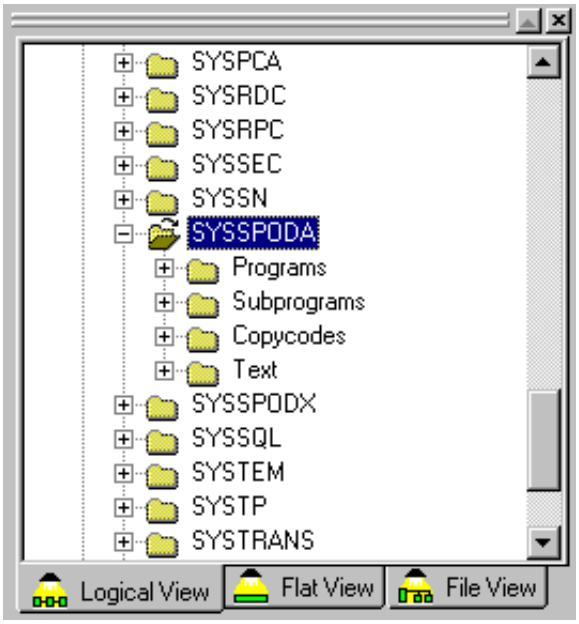
You will now log on to the library SYSSPODA which contains the objects that will be used in this tutorial.

To log on to a library

1. Make sure that logical view is active in your library workspace.
2. Under the node for the development server session that you have just mapped, click the plus sign next to "System Libraries".
   The system libraries are now listed.



3. Scroll down the list until the library SYSSPODA is shown in the tree.
4. To log on to the library, simply click the library name SYSSPODA.
   The status line at the bottom of the Natural Studio window informs you that this operation has been sent to the server.
5. To display the contents of the library, click the plus sign next to the library name SYSSPODA to expand the tree.



You can now proceed with the next exercise: Issuing Commands.

Issuing Commands

Natural Studio commands are usually issued via context menus as explained in this section. Several important context menus are shown, and copying and moving via drag-and-drop is explained.

Natural Studio also provides a command line in which you can directly enter Natural system commands. The prerequisite is that a certain logical context is given. For example, the SAVE command can only be executed when a source is currently shown in the editor.

A graphical user interface is not provided for all system commands that are available on the mainframe. When you issue such a command in the command line, terminal emulation will be started in a separate window, showing the corresponding character screen. You can then work in the same way as on the mainframe.

Certain system commands (for example, EDT) are not available in Natural Studio and can therefore not be executed from the command line.

For further information on system commands, refer to the Natural Reference documentation.

The following topics are covered below:

- Context Menus
- Creating User Libraries
- Copying and Moving Objects
- Deleting Objects
- Cataloging Objects
- Displaying the Last Commands
- Listing Objects
- Invoking Terminal Emulation

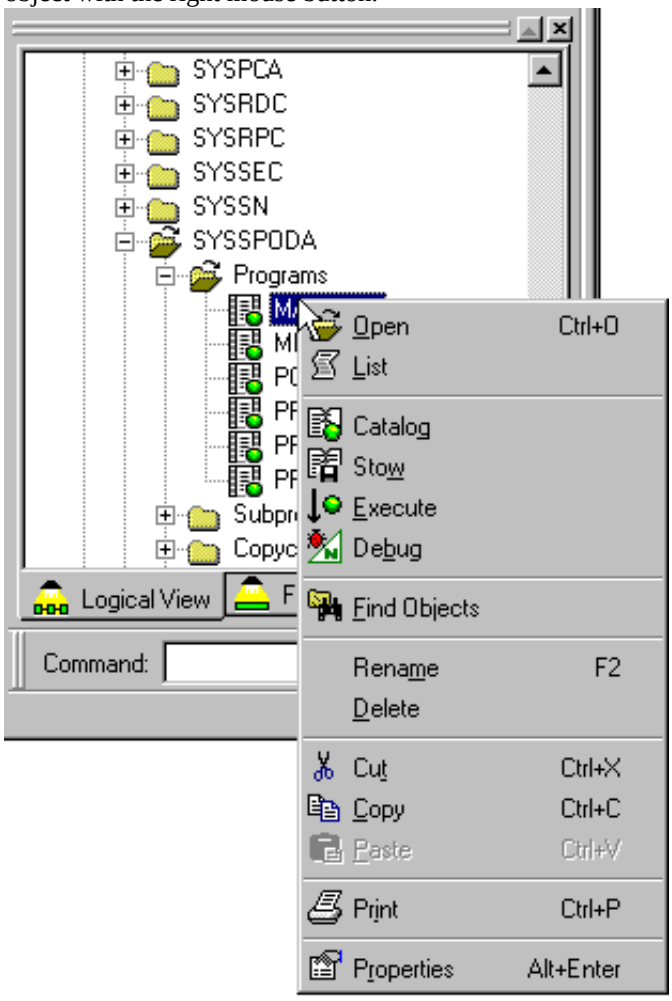
Context Menus

Context menus are invoked using the **right** mouse button. The commands provided in the context menu depend on the object or the position within the Natural Studio window that has been clicked.

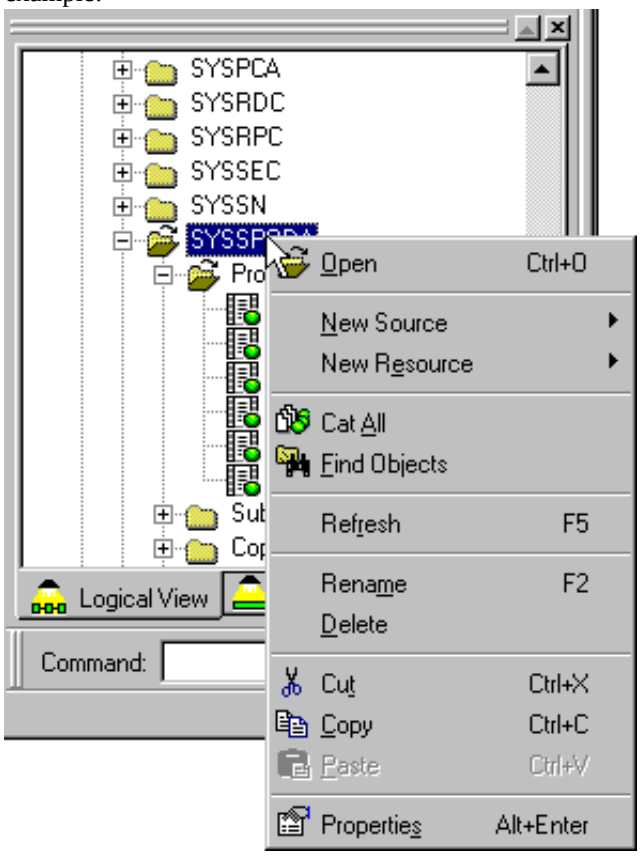
Note:
The menu bar at the top of the Natural Studio window can be customized. Thus when a menu is not shown in the menu bar, you can still issue the commands that apply to the selected object from the context menu.

When you click an element in a tree with the right mouse button, all valid commands for this element are shown in a context menu.

- The following example shows a context menu that has been invoked by clicking the name of a Natural object with the right mouse button.

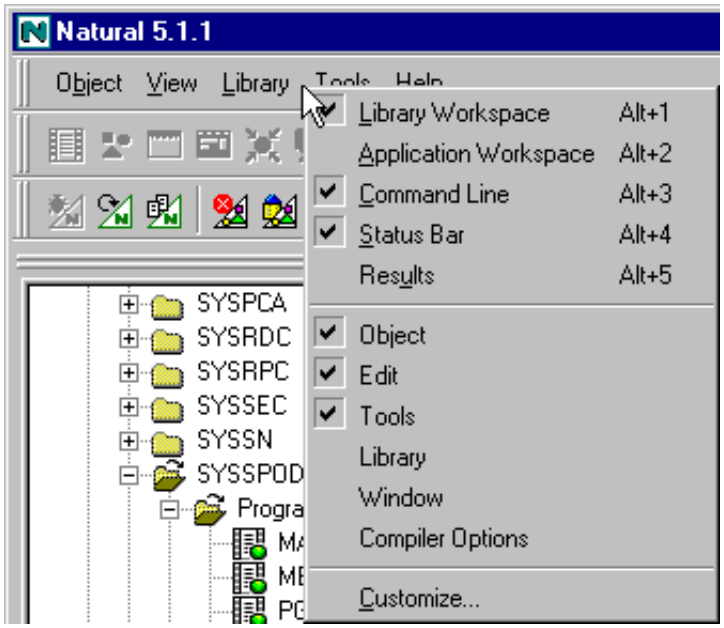


- A different context menu is shown when you click the name of a library with the right mouse button. For example:

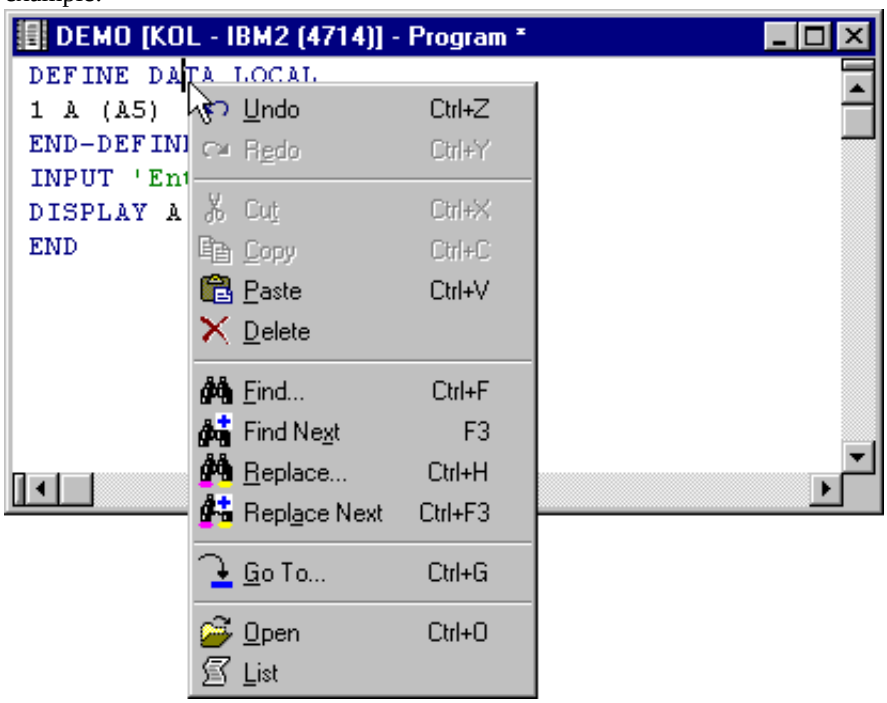


When you click any other position outside the library workspace with the right mouse button, different commands are shown in the context menu.

- The following example shows a context menu has been invoked by clicking the menu bar with the right mouse button:



- A different context menu is shown when you click the right mouse button in a program editor window. For example:



Note:
When a toolbar button or shortcut key is available, this information is shown in the context menu. Commands that are dimmed, are currently not available.

Creating User Libraries

You will now create the following user libraries that will be used later in this tutorial:

- SPODLIB
- SPODADD
- SPODTEST

Make sure to create these libraries in an environment in which a Natural development server has been installed.

To create the user libraries

1. Depending on the current view, you have to select one of the following (below the node of the development server to which you have previously connected):

Logical view:	"User Library"
Flat view:	"FUSER"
File view:	the name of the database

2. From the **Library** menu, choose **New**.
   Or click the right mouse button and from the resulting context menu, choose **New**.
   A new library with the default name USRNEW is now shown in the tree.
   

   The default name is selected so that you can immediately enter a new name. Any text you enter automatically deletes the selection.
3. Specify SPODLIB as the name of the library.
4. Press ENTER
   Or click any other position in the library workspace.
5. Create the libraries SPODADD and SPODTEST as described above.

Copying and Moving Objects

With the following exercises, you will

- copy the contents of the system library SYSSPODA to your user library SPODLIB,
- move all subprograms from library SPODLIB to library SPODADD,
- copy the program PGMCHECK from library SPODLIB to library SPODTEST.

These objects will be used later in this tutorial.

Different methods can be used to copy and move objects:

- menu commands (see **Copy** and **Paste** in the first exercise in this section)
- shortcut keys (see CTRL+C and CTRL+V the first exercise in this section)
- toolbar buttons (see  and  in the first exercise in this section)
- drag-and-drop (see the last two exercises in this section)

To copy the contents of the system library SYSSPODA to the user library SPODLIB

1. Select the system library SYSSPODA.
2. Click the right mouse button and from the resulting context menu, choose **Copy**.
   Or press CTRL+C.
   Or click the following toolbar button:
   
   You can now paste the contents of the library to your user library.
3. Scroll to the user library SPODLIB that you have previously created.
4. Select the user library SPODLIB.
5. Click the right mouse button and from the resulting context menu, choose **Paste**.
   Or press CTRL+V.
   Or click the following toolbar button:
   
   All objects from the system library SYSSPODA are now copied to your user library SPODLIB.

To move all subprograms from the library SPODLIB to the library SPODADD using drag-and-drop

1. Make sure that logical view is active.
2. Click the plus sign next to the library SPODLIB to expand the tree.
3. Click the "Subprograms" node and hold down the mouse button.
4. Drag the selected object to the name of the node SPODADD.
5. Release the mouse button.
   All subprograms are now moved to the library SPODADD.

To copy the program PGMCHECK from the library SPODLIB to the library SPODTEST using drag-and-drop

1. Make sure that flat view is active.
   Thus you need not open the folder containing all programs which is provided in logical view.
2. Under the node FUSER, click the plus sign next to the library SPODLIB to expand the tree.
3. Click the program PGMCHECK and hold down the mouse button.
4. Hold down CTRL.
5. Drag the selected object to the name of the node SPODTEST.
6. Release the mouse button and then CTRL.
   The program is now copied to the library SPODTEST.

Note:
When you move an object, you cut it at its original position and paste it at a new position. To copy an object, press CTRL while dragging. This does not cut the object at its original position.

Deleting Objects

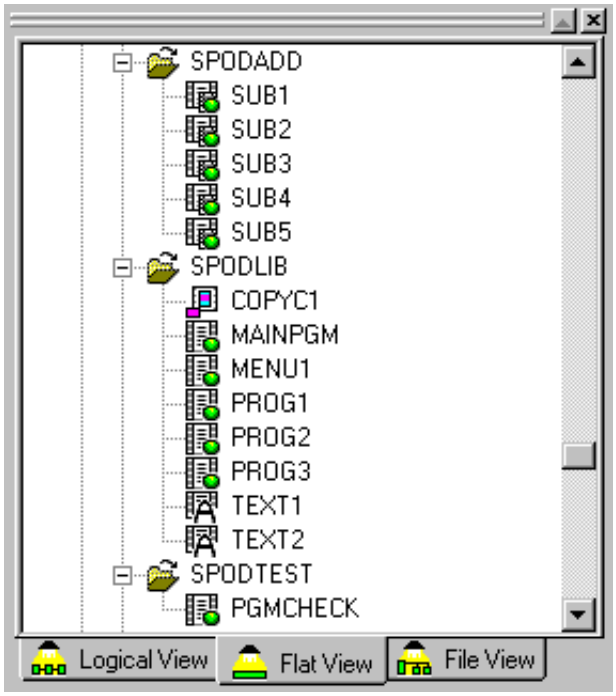
Since you have copied the program PGMCHECK in the previous exercise, it is available in two libraries. You will now delete this program from the library SPODLIB. This exercise assumes that flat view is still active.

To delete the program PGMCHECK from the library SPODLIB

1. Under the node for the library SPODLIB, select the program PGMCHECK.
2. From the **Object** menu, choose **Delete**.
   Or click the right mouse button and from the resulting context menu, choose **Delete**.
   A dialog box appears, asking whether you really want to delete the program.
3. Choose the **Yes** button to delete the program.

Note:
You can switch off display of delete messages. From the **Tools** menu, choose **Options**. In the resulting dialog box, display the "Workspace" tab, deselect the "Display delete messages" check box and choose the **OK** button.

With all nodes expanded in flat view, your new libraries should now look as follows:



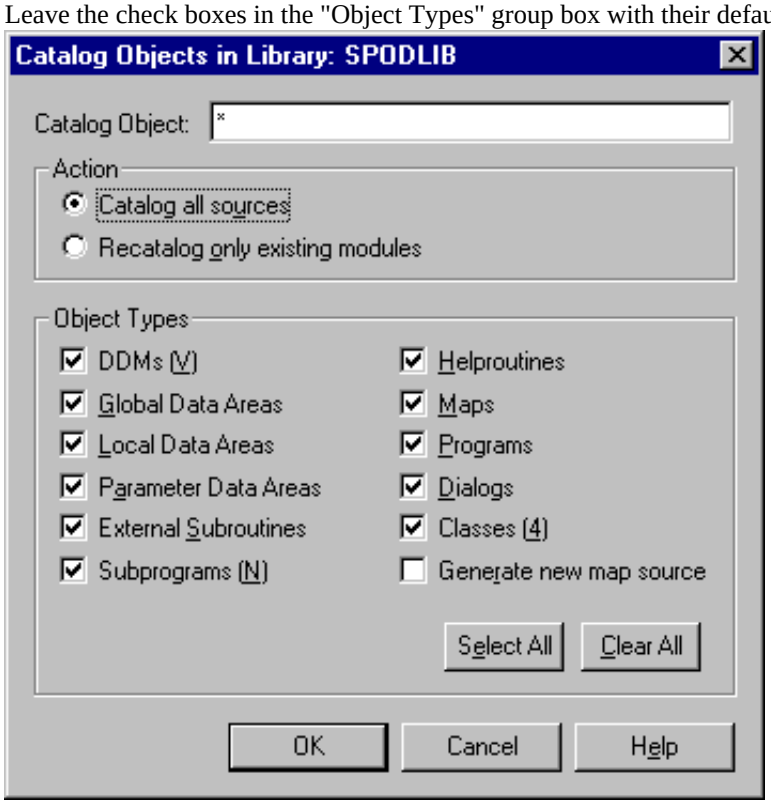
Cataloging Objects

CATALL is one of the Natural commands for which a graphical user interface is provided in Natural Studio.

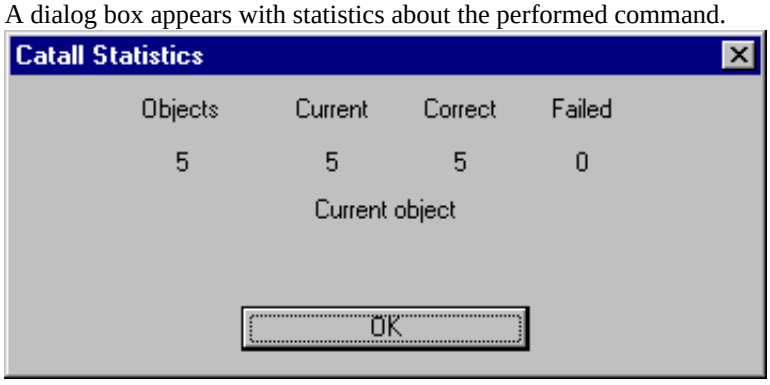
You will now catalog the objects of the libraries you have just created.

To catalog all objects in a library

1. Select the library SPODLIB.
2. From the **Library** menu, choose **Cat All**.
3. Or click the right mouse button and from the resulting context menu, choose **Cat All**.
   The "Catalog Objects in Library: *libraryname*" dialog box appears.
4. Make sure that the option button "Catalog all sources" is selected.
   Leave the check boxes in the "Object Types" group box with their default settings.



5. Choose the **OK** button.
   A dialog box appears with statistics about the performed command.



6. Choose the **OK** button to close the dialog box.
   A dialog box appears, informing you that cataloging was successful.

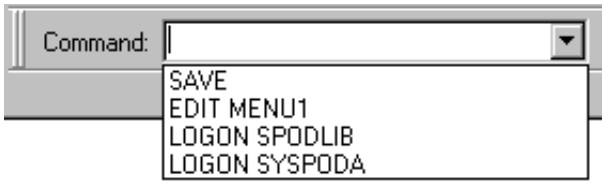
7. Choose the **OK** button to close the dialog box.
8. Repeat the above steps for the libraries SPODADD and SPODTEST.

Note:
You can switch off display of success messages. From the **Tools** menu, choose **Options**. In the resulting dialog box, display the "Workspace" tab, deselect the "Display success messages" check box and choose the **OK** button.

Displaying the Last Commands

Natural commands can be executed from the drop-down list box of the command line.

Natural Studio saves each character string you enter in the command line for the current session. The drop-down list box contains your last entries. You can select an entry and press ENTER to execute it once more.



When you enter the first character of a command that you have previously entered, the corresponding command is automatically copied to the command line. In this case, you just have to press ENTER to execute it.

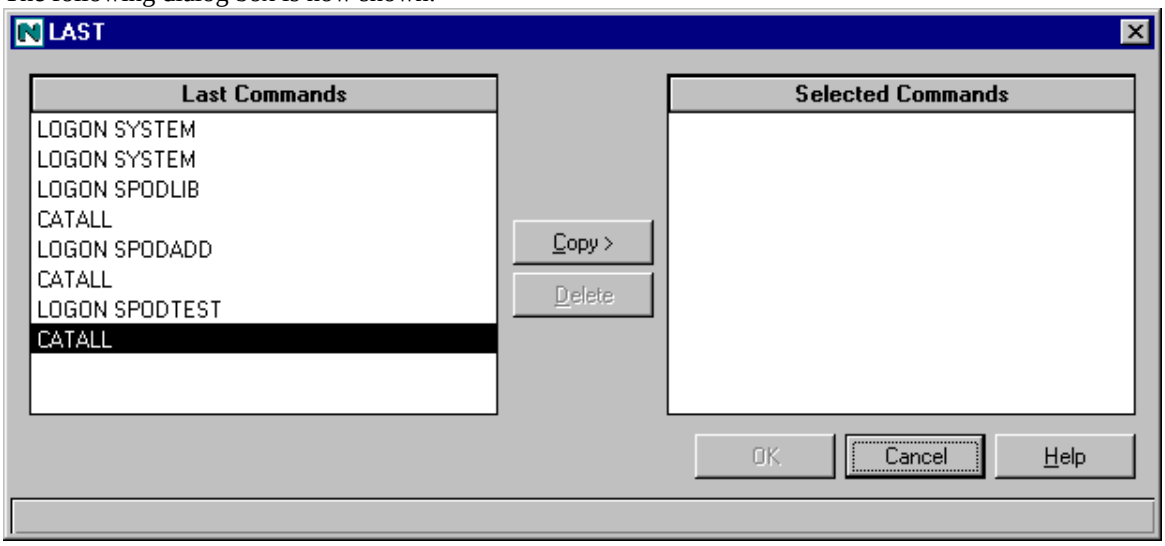
When the mouse pointer is positioned on the command line, you can use the right mouse button to invoke a context menu. Using the commands from this context menu, you can, for example, paste a text string in the command line.

The LAST * command is an example of a Natural command for which a menu command is not provided with Natural Studio. In contrast to the commands that you enter in the command line, the LAST * command displays all system commands in the order in which they were entered in a dialog box. It considers all Natural commands that were issued during a session: for example, when you click a library to log on to it, or when you execute a program using a menu command, toolbar button or shortcut key. Furthermore, it allows you to select several commands to be executed one after the other (see below).

If the command line is not shown in the Natural Studio window, display it as described previously in Displaying the Command Line.

To execute the LAST * command from the command line

1. Enter the following command in the command line and press ENTER:
   LAST *
   The following dialog box is now shown:



2. On the left, select the first command that you want to execute.
3. Choose the **Copy** button.
   The selected command is now shown on the right.
4. Optionally. Modify the command on the right (for example, specify another library name with the LOGON command).
5. Repeat the above steps to copy all commands that you want to execute to the right.
   The commands will be executed in the same sequence in which they appear in the list.
6. Choose the **OK** button to execute all selected commands one after the other.

Listing Objects

You will now list the objects in one of the libraries you have previously created. You can do this in two different ways:

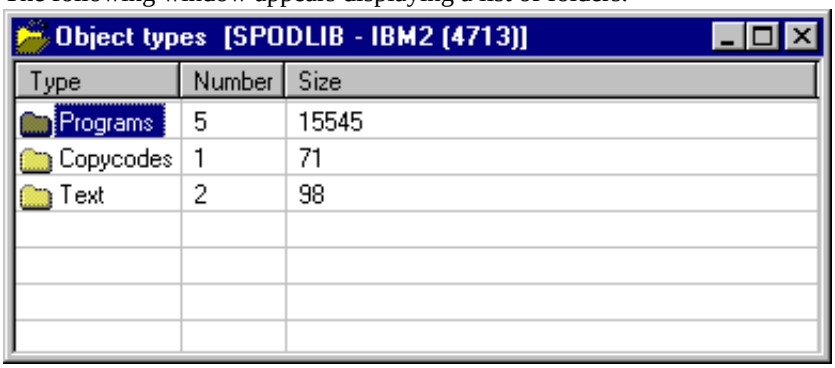
- choose **Open** from the context menu, or
- enter the `LIST *` command in the command line

When using the **Open** command, the contents of the window depends on the current view. For example, in logical view, all folders are shown, and in flat view all objects are shown.

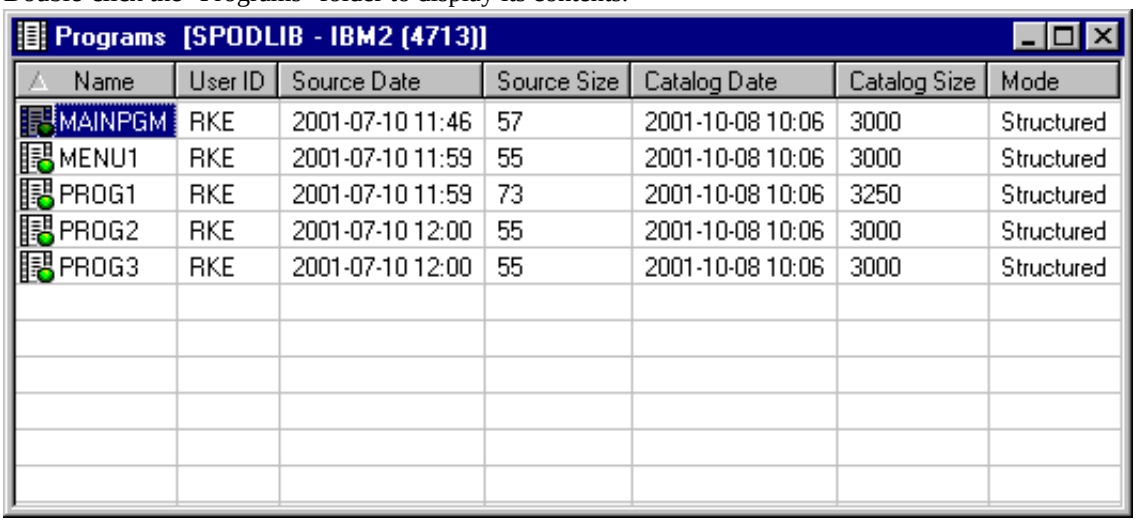
When entering `LIST *` in the command line, all objects of a library are shown. The contents of the resulting window is always the same. The current view is not considered. An advantage of the `LIST` command is that you can also specify that only objects starting with a specific letter are shown (e.g. `LIST P*`).

To list all objects in a library with the Open command

1. Make sure that logical view is active for this exercise.
2. Select the library SPODLIB in the library workspace.
3. Click the right mouse button and from the resulting context menu, choose **Open**.
   The following window appears displaying a list of folders.



4. Double-click the "Programs" folder to display its contents.



5. To change the display sequence, click the column header "Source Date".
   An arrow is now shown in this column header indicating the current display sequence.
6. Click the column header "Source Date" once more.
   This toggles between ascending and decending display sequence.
7. Close each of these two windows by clicking the standard close button at the top right of a window.

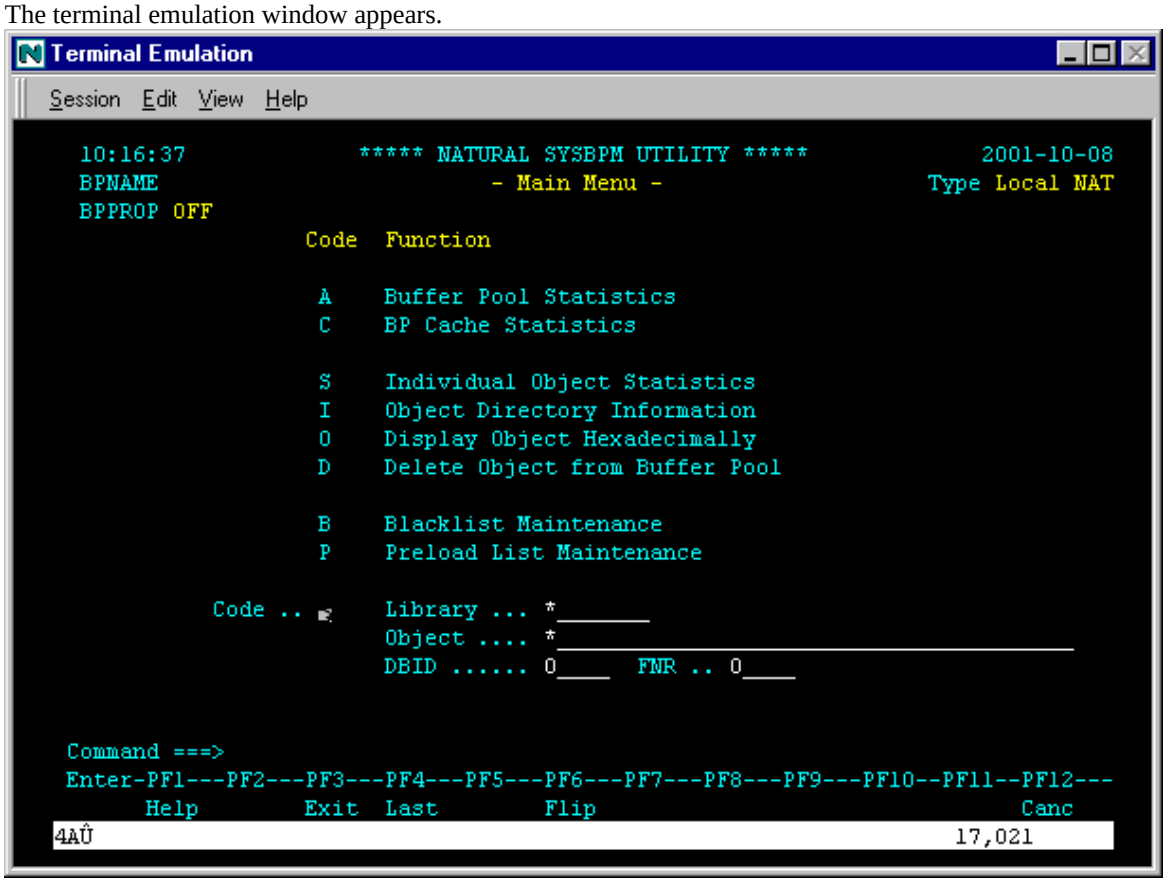
Invoking Terminal Emulation

SYSBPM is a Natural utility for which a graphical user interface is not provided in Natural Studio. When you invoke this utility, its character-based mainframe screen is shown in a terminal emulation window.

Note:
As long as terminal emulation is active, you cannot work with Natural Studio.

To invoke the SYSBPM utility in a terminal emulation window

1. Enter the following command in the command line:
   SYSBPM
   The terminal emulation window appears.



   You can now use the utility in the same way as on the mainframe. Command buttons are provided for frequently-used PF keys. When they are not shown, you can display them as described with the following step.
2. From the **View** menu, choose a PF-key set (for example, **PF Keys 1-12**) to display the command buttons for the PF keys.
3. To terminate terminal emulation, issue the EXIT command (either by issuing it in the command line or by pressing the corresponding PF key).

You can now proceed with the next exercise: Handling Programs.

Handling Programs

This section shows the most important steps for handling programs, including testing and debugging.

For more information, see Program Editor in the Natural for Windows User's Guide.

For detailed information on the debugger and remote debugging, see Debugger in the Natural for Windows documentation.

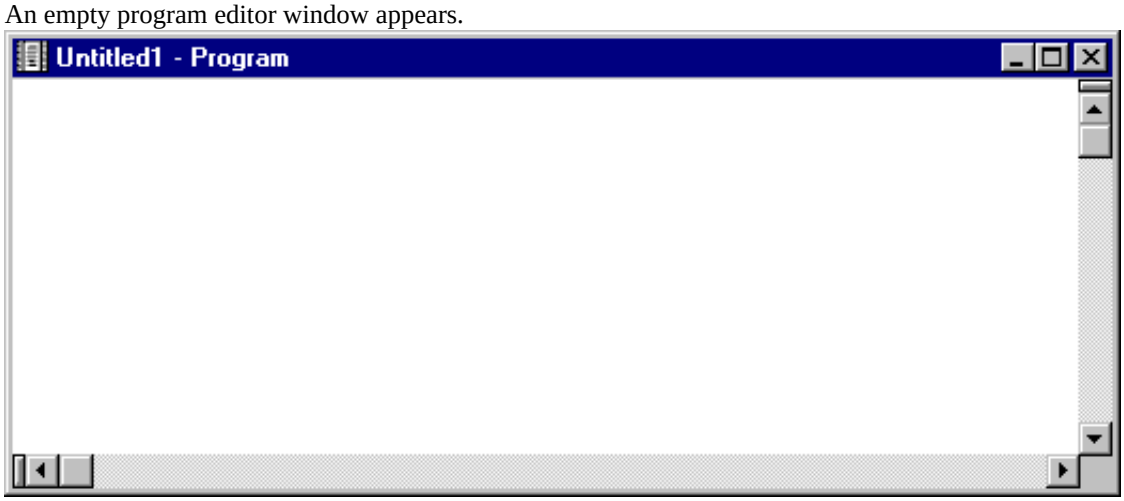
The following topics are covered below:

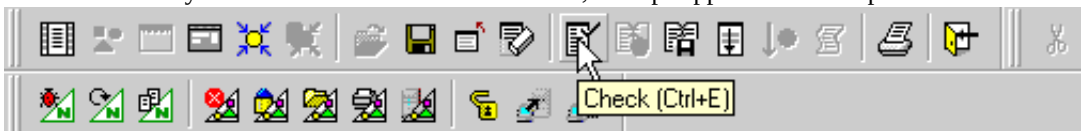
- Creating a New Program
- Stowing a Program
- Executing a Program
- Debugging a Program

Creating a New Program

You will now create a small program which prompts the user for input and displays this input. You can create this program in any library that you are allowed to use.

To create a program

1. Click the name of the library in which you want to store the new program.
2. From the **Object** menu, choose **New > Program**.
   Or click the right mouse button and from the resulting context menu, choose **New Source > Program**.
   Or press CTRL+N.
   Or click the following toolbar button:
   
   An empty program editor window appears.
   

   The status line at the bottom of the Natural Studio window now shows additional information (line and column in which the cursor is currently positioned and the current size of the program).
   When the program editor window is the active window, all toolbar buttons that apply to a program are activated. When you move the mouse over the toolbar, tooltips appear. For example:
   

3. In the program editor, enter the following code:

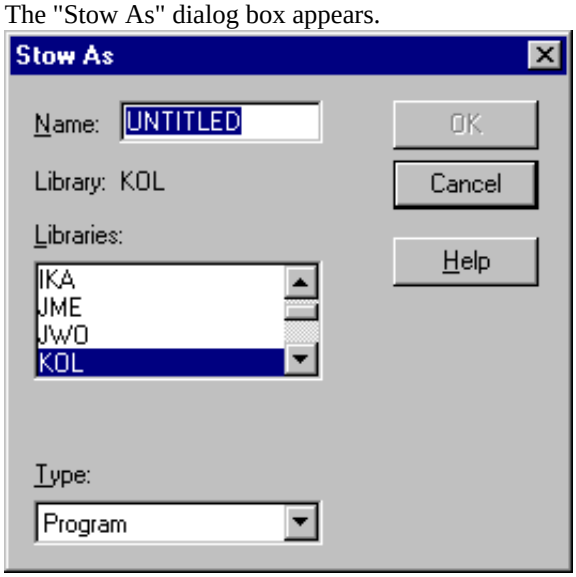
```
DEFINE DATA LOCAL
1 A (A5)
END-DEFINE
INPUT 'Enter' A
DISPLAY A
END
```


Stowing a Program

You will now stow the program you have just created.

Since it is possible to edit several programs at the same time (this is not possible on the mainframe), you have to make sure that the editor window for the program that you want to stow is the active window. To activate a window, you simply have to click it.

To stow a program

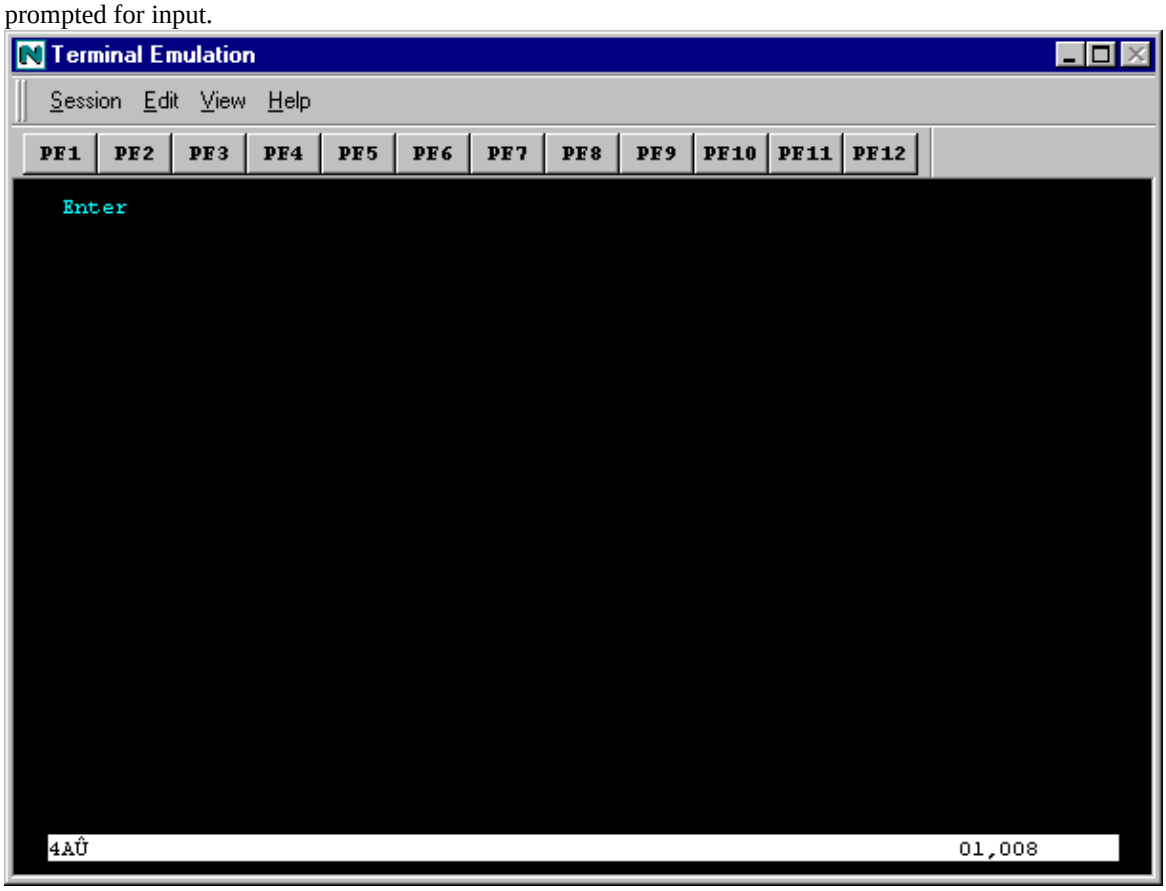
1. Make sure that the editor window containing your new program is active.
2. From the **Object** menu, choose **Stow**.
   Or click the following toolbar button:
   
   The "Stow As" dialog box appears.
   

3. In the "Name" text box, enter "DEMO" as the name of your program.
4. To stow the program in the current library, choose the **OK** button.
   If you have not switched off the success messages, a dialog box appears, informing you that the stow operation was successful. When this dialog box appears, press ENTER to close it.
   The new program is now shown in the library workspace. The title bar of the program editor window now shows the program name, the library in which it is stored, and the name and port number of the current development server.
5. Close the program editor by clicking the standard close button at the top right of the program editor window.

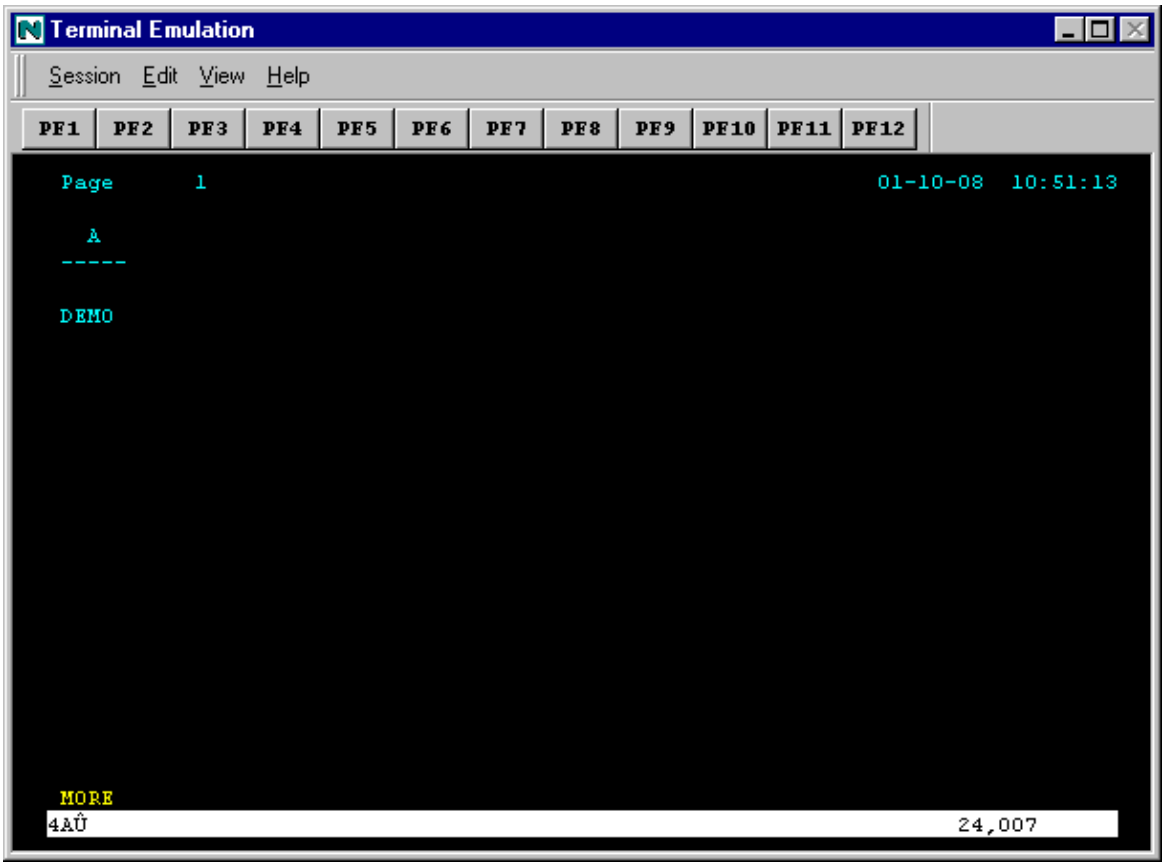
Executing a Program

You will now execute your DEMO program.

To execute a program

1. In the library workspace, select the program DEMO.
2. Click the right mouse button and from the resulting context menu, choose **Execute**.
   Or click the following toolbar button:
   
   The INPUT statement from the above example program invokes a terminal emulation window. You are prompted for input.
   

3. Enter the string "Demo" at the "Enter" prompt and press ENTER.
   The DISPLAY statement from the above example program now shows the string you have just entered.



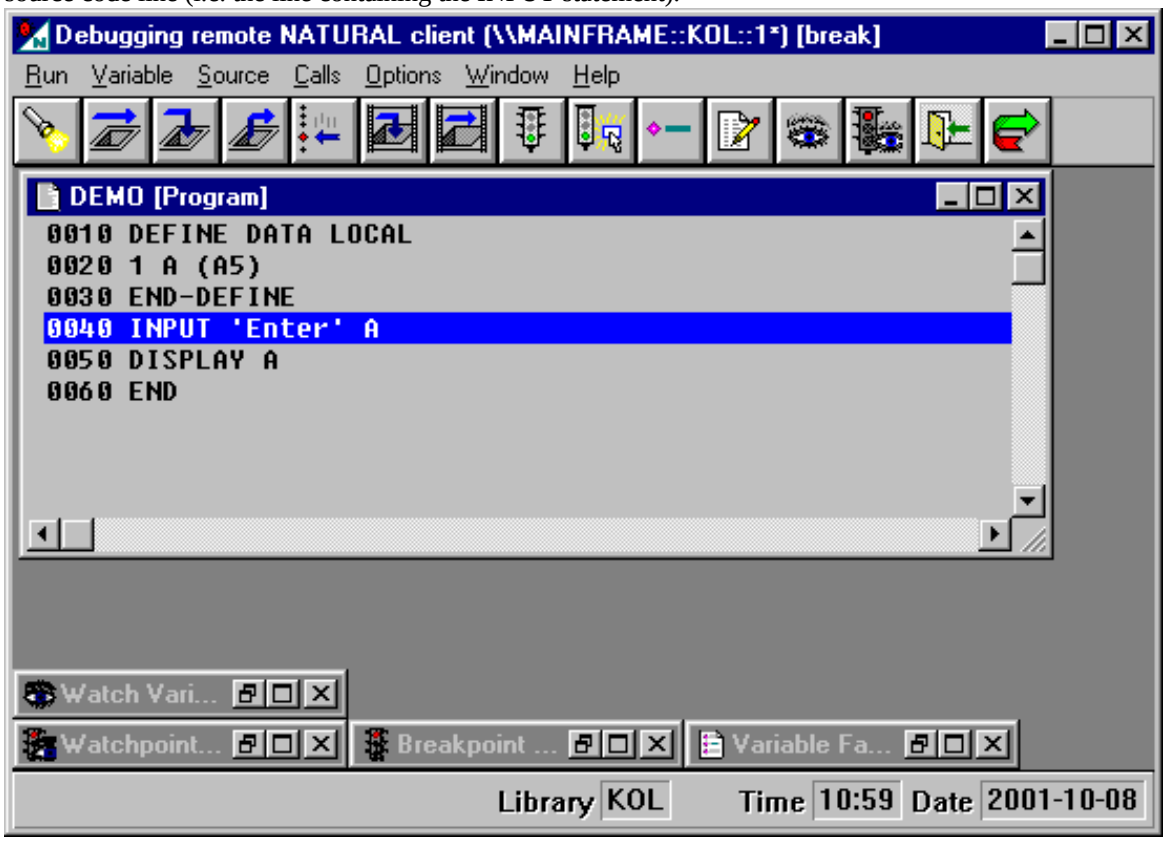
4. Press ENTER.
   The example program is ended and the terminal emulation window is automatically closed.

Debugging a Program

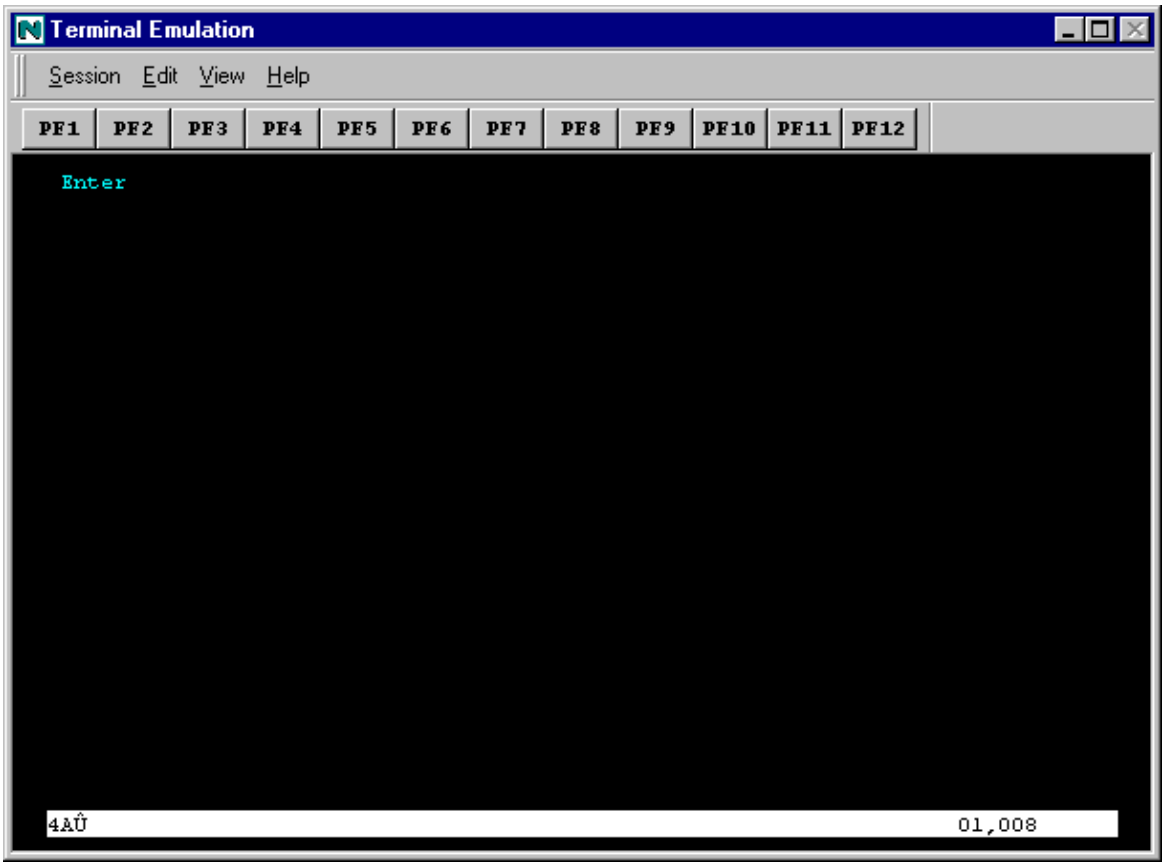
You will now debug your DEMO program.

Note:
As long as the debugger is active, you cannot work with Natural Studio.

To debug a program

1. In the library workspace, select the program DEMO.
2. From the **Tools** menu, choose **Development Tools > Debugger**.
   Or click the right mouse button and from the resulting context menu, choose **Debug**.
   Or click the following toolbar button:
   
   The Natural Debugger is invoked in a window of its own. The trace bar is positioned in the first executable source code line (i.e. the line containing the INPUT statement).
   

3. From the **Run** menu, choose **Animated step into**.
   Or press CTRL+A.
   Or click the following toolbar button:
   
   Note:
   Similar toolbar buttons are available for the commands **Step into** and **Animated step into**. Make sure to click the correct toolbar button. When you move the mouse pointer over the toolbar buttons, descriptions for the buttons are shown in the status bar.
   The program is now executed step by step until the end of the program. The debugger steps through any Natural object invoked or included.
   The INPUT statement invokes a terminal emulation window.



Note:
It may happen that the terminal emulation window is hidden behind the debugger window.

4. Make sure that the terminal emulation window cannot be overlapped by the debugger window.
   Otherwise you cannot see the result of the DISPLAY statement (see below) in the terminal emulation window.
5. Enter the string "Demo" in the terminal emulation window and press ENTER.
   The debugger window is shown again. The trace bar is positioned in the next executable source code line.
   The DISPLAY statement shows the string you entered in the terminal emulation window. The debugger then proceeds to the last line and displays the message that the debugging session has terminated.
6. Press ENTER to close the message box.
   The debugger window is also closed. The terminal emulation window is still shown.
7. To close the terminal emulation window, press ENTER

You can now proceed with the next exercise: Locking and Unlocking.

Locking and Unlocking

An object locking mechanism prevents concurrent updates when working with objects that are stored on a remote development server.

The exercises below demonstrate locking for an object that you are currently modifying in the program editor. You will map to the same development server once more. When you then try to open the same object from the new session, a locking message is shown.

For more information, see Object Locking in the Remote Development documentation of Natural for Windows.

The following topics are covered below:

- Opening an Object
- Opening the Same Object from Another Session
- Unlocking Objects
- Displaying a List of All Locked Objects
- Moving Folders Containing Locked Objects

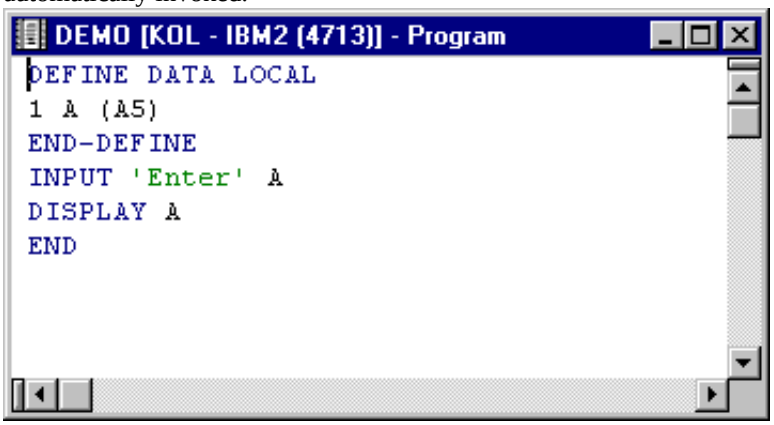
Opening an Object

When you open an object, it is locked for all other users until you close it.

You will now display the Natural code of the DEMO program that you have previously created. It is assumed that logical view is active.

To open the DEMO program in logical view

1. Under the node for the user library in which you have stored the DEMO program, click the plus sign next to the "Programs" folder.
   This expands the node and shows all programs in this library.
2. Click the name of the DEMO program with the right mouse button and from the resulting context menu, choose **Open**.
   Or press CTRL+O.
   Or double-click the program name.
   The program is now shown in the source area of the Natural Studio window. The corresponding editor is automatically invoked.

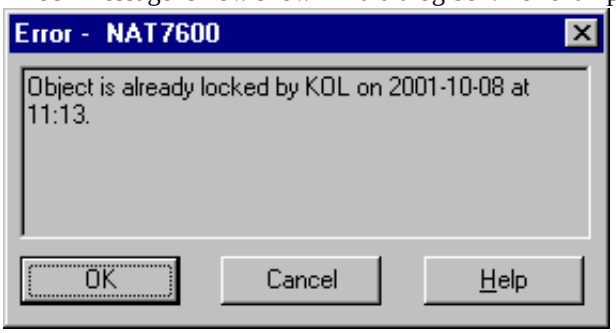


   Do not close this program editor window. Leave it open so that locking can be demonstrated with the next exercise.

Opening the Same Object from Another Session

When you try to open an object that is currently being modified by another user, the corresponding lock message is shown. This message tells you which user is currently modifying the object and the date and time it was locked.

To map the same development server once more and try to open a locked program

1. From the **Tools** menu, choose **Map Environment**.
   Or click the following toolbar button.
   
   The "Map Environment" dialog box appears.
2. Proceed as described previously under Connecting to a Development Server for the First Time.
3. In the new session, expand the node of the library containing the DEMO program.
4. Double-click the program name DEMO.
   A lock message is now shown in a dialog box. For example:
   

5. Choose the **OK** button to close the dialog box.

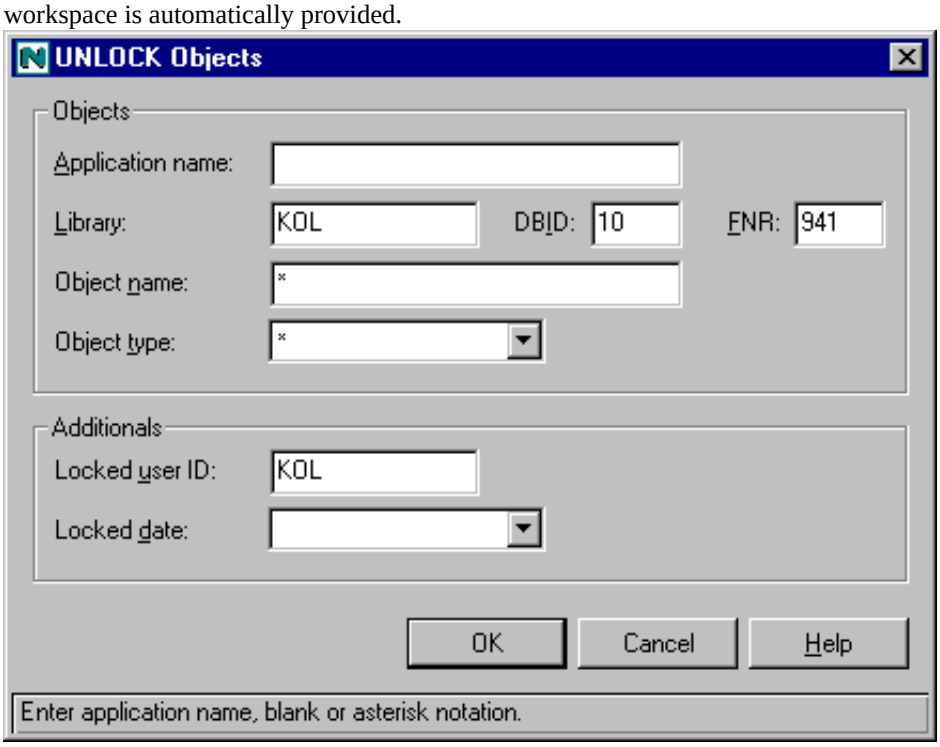
Unlocking Objects

You will now unlock your DEMO program from the session in which it is currently locked (not the session in which it was opened).

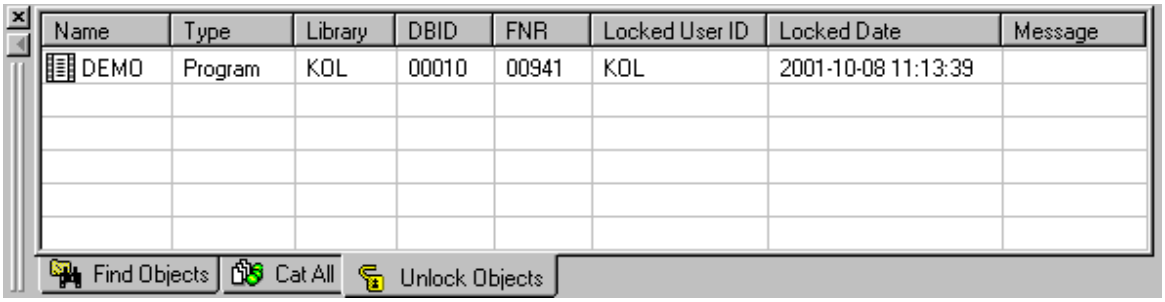
Note:
When Natural Security is active, it is possible that you cannot use certain commands. Thus, it may be possible that you are not allowed to unlock your own objects.

To unlock a locked object

1. In library workspace, select the library containing the locked DEMO program.
2. From the **Tools** menu, choose **Development Tools > Unlock Objects**.
   Or click the following toolbar button:
   
   The "Unlock Objects" dialog box appears. The name of the library that is currently selected in library workspace is automatically provided.



3. Choose the **OK** button (without entering any information).
   The results window is now shown in the Natural Studio window.



4. In the results window, click the object to be unlocked with the right mouse button.
5. From the resulting context menu, choose **Unlock Objects**.
   The object is still shown in the results window. The "Message" column, however, indicates that the object has been unlocked.

6. To hide the results window, click the standard close button at the top of the results window.
   Or from the **View** menu, choose **Results**.
   Or press ALT+5.
   When the results window is not shown in the Natural Studio window, a check mark is not shown in the **View** menu next to the **Results** command.

Note:
You can undock and dock the results window as described previously in this tutorial.

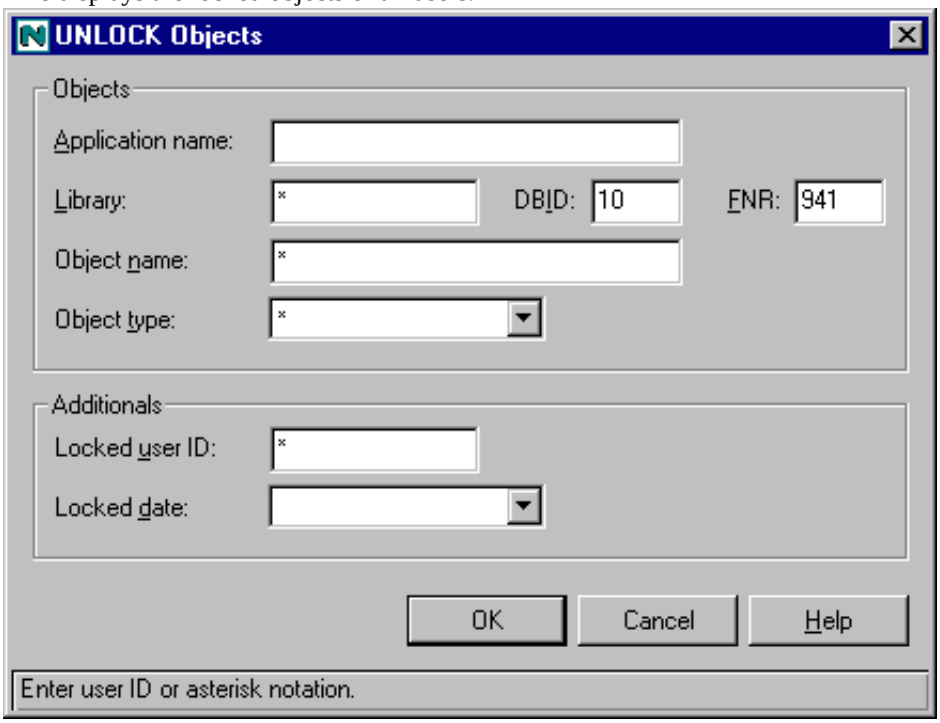
Displaying a List of All Locked Objects

You can display a list of all locked objects for the currently active development server. This includes the objects of all users in all libraries. You can unlock any object contained in this list.

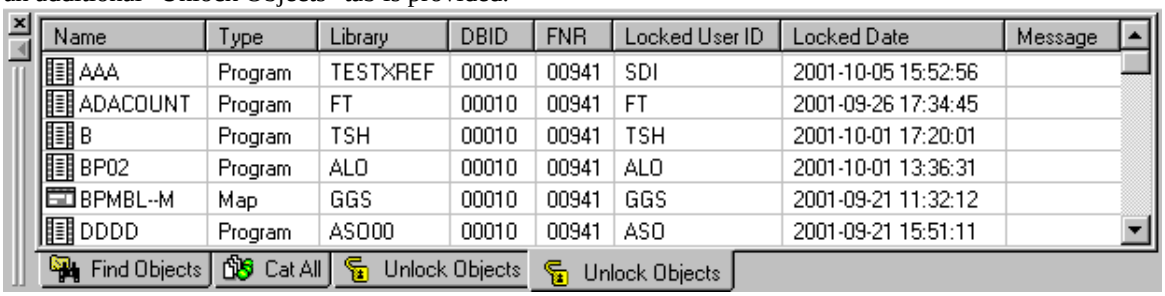
It is only possible to display and unlock objects from another user in a non-secure environment (i.e. when Natural Security is not active on the development server). In a secure environment, the administrator defines which user locks may be unlocked by other users.

To display all locked objects

1. From the **Tools** menu, choose **Development Tools > Unlock Objects**.
   Or click the following toolbar button:
   
   The "Unlock Objects" dialog box appears.
2. Enter an asterisk in the "Library" text box.
   This displays the locked objects in all libraries.
3. Enter an asterisk in the "Locked user ID" text box.
   This displays the locked objects of all users.



4. Choose the **OK** button.
   The results window is shown again. Since this is the second time you issued the **Unlock Objects** command, an additional "Unlock Objects" tab is provided.



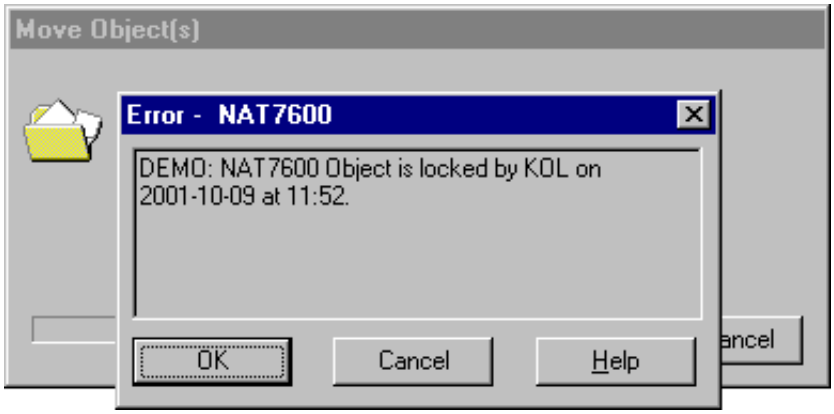
5. If you want to delete the first "Unlock Objects" tab, click it to display its contents.

6. Click an object in the "Name" column with the right mouse button and from the resulting context menu, choose **Delete Tab**.
7. Before you continue with the next section, hide the results window as described above.

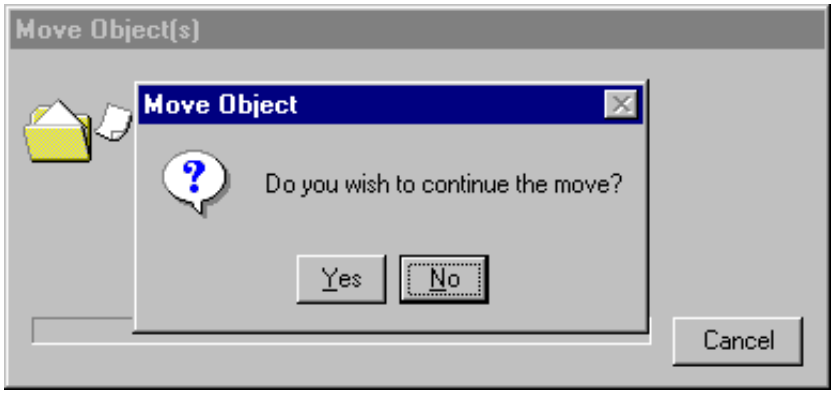
Moving Folders Containing Locked Objects

Locked objects cannot be moved.

When you try to move a folder containing locked objects to another folder (for example, via drag-and-drop), a lock message appears.



Choose the **OK** button to close the dialog box. Another dialog box appears, asking whether you want to continue the move.



When you choose the **Yes** button, all objects except the locked objects are moved.

You can now proceed with the next exercise: Handling Applications.

Handling Applications

An application is a collection of Natural objects and non-Natural objects which build a functional unit from the business point of view.

There are two types of applications:

- **Base application**
  A set of Natural objects that are stored in the same user system file (FUSER). You can link objects of different libraries to a base application.
- **Compound application**
  You can link several base applications to a compound application. A compound application can be spread across different hosts.

For detailed information, see Application Concept in the SPoD documentation.

The following topics are covered below:

- Prerequisites
- Displaying the Application Workspace
- Creating a Base Application
- Creating a Compound Application
- Linking Objects to a Base Application
- Linking Base Applications to a Compound Application
- Managing Linked Objects
- Mapping an Application
- Displaying the Properties of an Application

Note:
The applications that you will create with the exercises below do not run. Their purpose is just to demonstrate the basics for handling applications.

Prerequisites

It is assumed for the following exercises that you have:

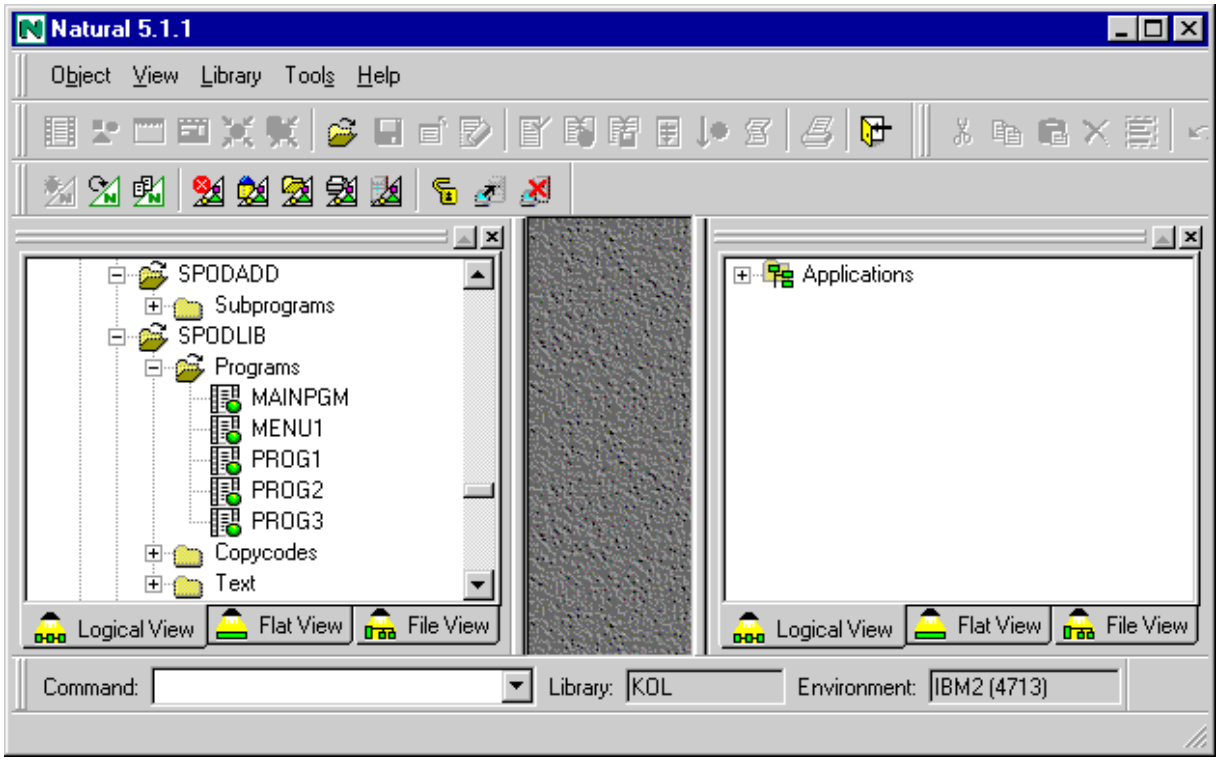
- copied the contents of the system library SYSSPODA to your user library SPODLIB,
- moved all subprograms from library SPODLIB to library SPODADD,
- copied the program PGMCHECK from library SPODLIB to library SPODTEST

as described previously in this tutorial.

Displaying the Application Workspace

The application workspace is the area in which all known applications can be displayed. It provides the same views as the library workspace (logical view, file view and flat view). Your application workspace is initially empty.

When you start Natural Studio for the first time, your application workspace is not shown. The exercise below explains how to display it. It is initially displayed at the right of the Natural Studio window.



To toggle application workspace display

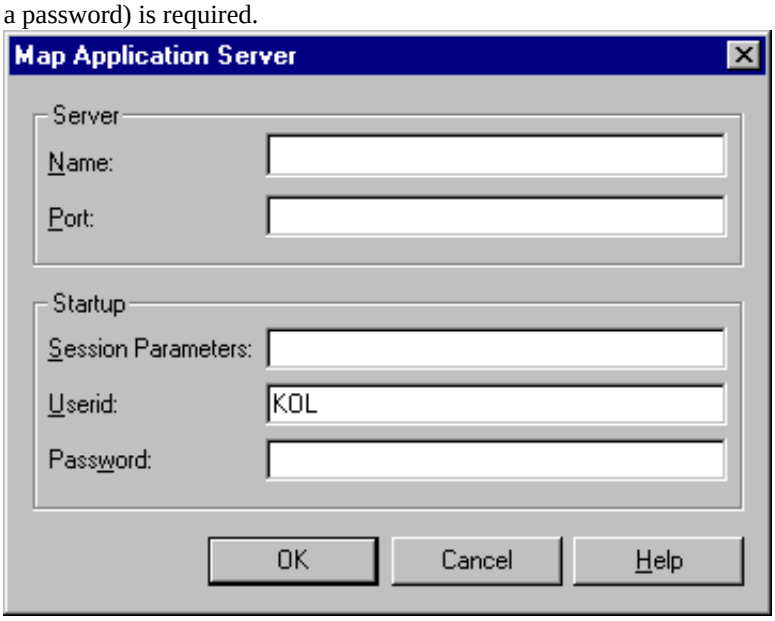
- From the **View** menu, choose **Application Workspace**.
  Or press ALT+2.
  When the application workspace is displayed in the Natural Studio window, a check mark is shown next to the **Application Workspace** command.

Creating a Base Application

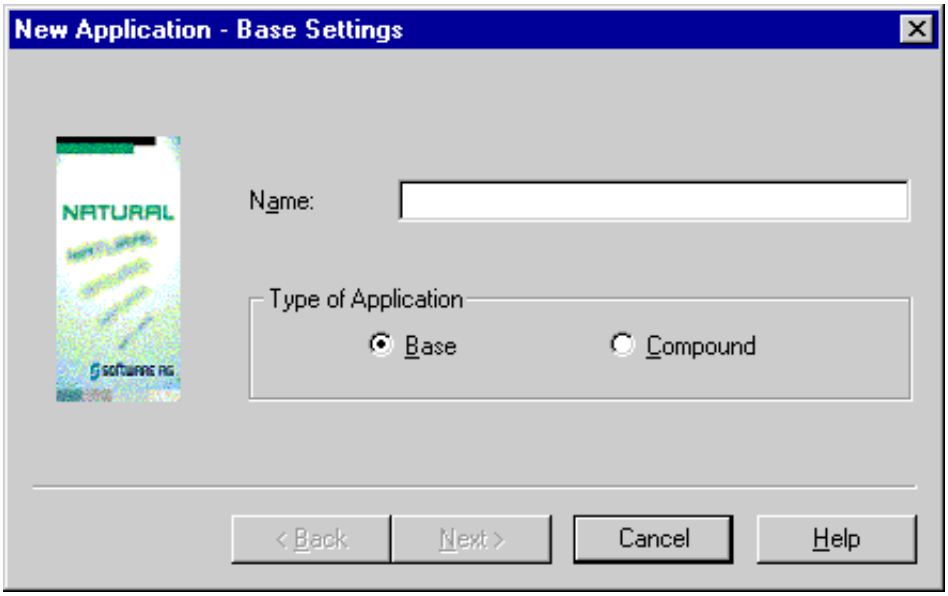
You will now create two base applications with the names SPODAPPL1 and SPODAPPL2. You will create them on the same development server that you have previously defined for library view.

To create a new base application

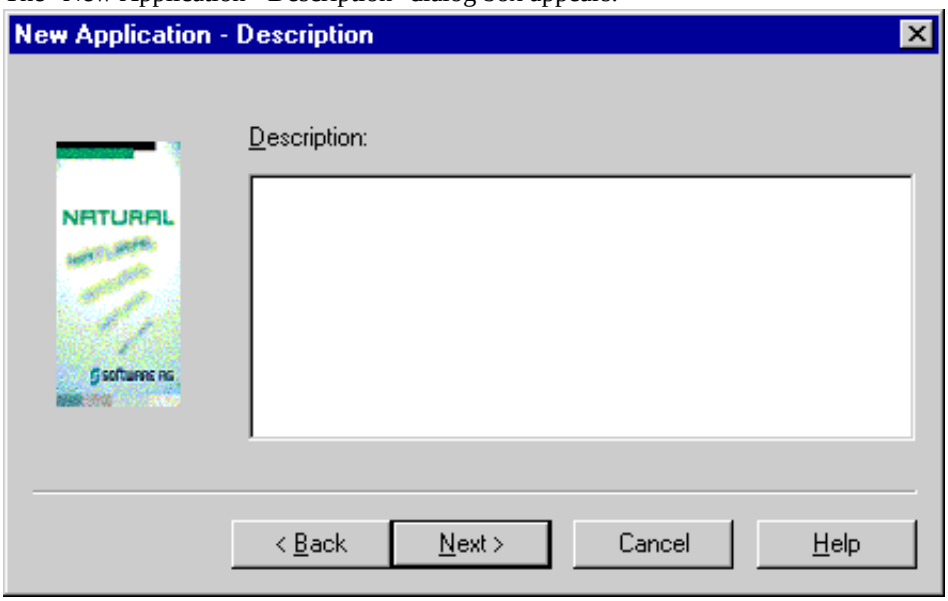
1. In the application workspace, click the node name "Applications" with the right mouse button.
2. From the resulting context menu, choose **New**.
   The "Map Application Server" dialog box appears. This dialog box appears when you work with applications for the first time. You have to enter the development server on which the Application Manager is located. A development server session will then be started for the connection to the Application Manager. When Natural is started the next time, the development server session for the Application Manager connection will be started automatically and this dialog box will appear only if additional information (e.g. a password) is required.



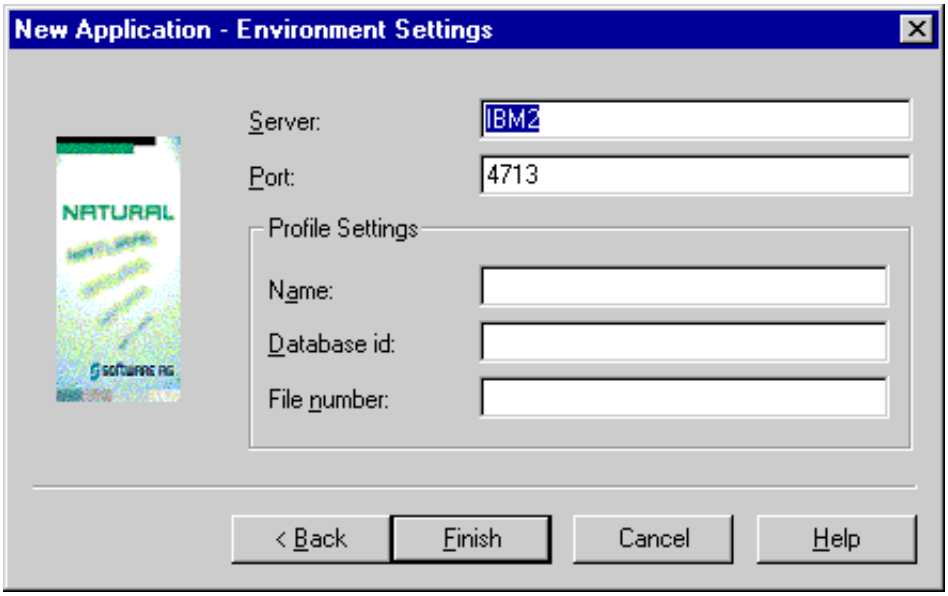
3. In the "Name" text box, enter the name of the development server on which the Application Manager is located.
4. In the "Port" text box, enter the port number of the development server.
5. If Natural Security is installed on the development server, specify the required password in the "Password" text box. Otherwise, leave this text box blank.
6. Choose the **OK** button.
   The "New Application - Base Settings" dialog box appears.



7. Enter the name SPODAPPL1 for your first base application.
8. Make sure that the "Base" option button is selected.
9. Choose the **Next** button.
   The "New Application - Description" dialog box appears.



10. Optionally. Enter a description for your application.
    This can be any text.
11. Choose the **Next** button.
    The "New Application - Environment Settings" dialog box appears.



Name and port number of the application server that you have previously mapped ("Map Application Server" dialog box) are provided as the default values for the new application. For this tutorial, we will use the default values. The application server is the development server where the Application Manager is located.

Note:

It is also possible to use another development server.

12. Optionally. Specify the profile settings (Name, DBID and FNR) to control the session settings as on the mainframe.
13. Choose the **Finish** button.

    Note:

    If a password is required, the "Map Application *applicationname*" dialog box appears. Server name, port number and session parameters cannot be changed in this dialog box. They are fixed for an application. If this dialog box appears, specify the password and choose the **OK** button.

    The new application is now mapped. It is shown in the application window. Each time you map an application, a new development server session is started for this application.
14. Repeat the above steps to create the second base application with the name SPODAPPL2.

    You will later link objects to these two base applications.

    Note:

    Since you have already defined the development server for the connection to the Application Manager, the "Map Application Server" dialog box is not shown for the second application.

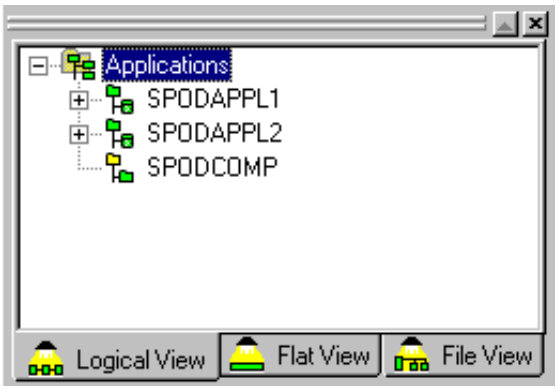
Creating a Compound Application

You will now create a compound application with the name SPODCOMP.

To create a new compound application

1. In the application workspace, click the node name "Applications" with the right mouse button.
2. From the resulting context menu, choose **New**.
   The "New Application - Base Settings" dialog box appears.
   Enter the name SPODCOMP for your compound application.
3. Make sure that the "Compound" option button is selected.
4. Choose the **Next** button.
   The "New Application - Description" dialog box appears.
5. Optionally. Enter a description for your application.
   This can be any text.
6. Choose the **Finish** button.
   You will later link your two base applications to this compound application.
   Note:
   If a password is required, a dialog box appears in which you have to specify the password.

Your application workspace should now look as follows:



Linking Objects to a Base Application

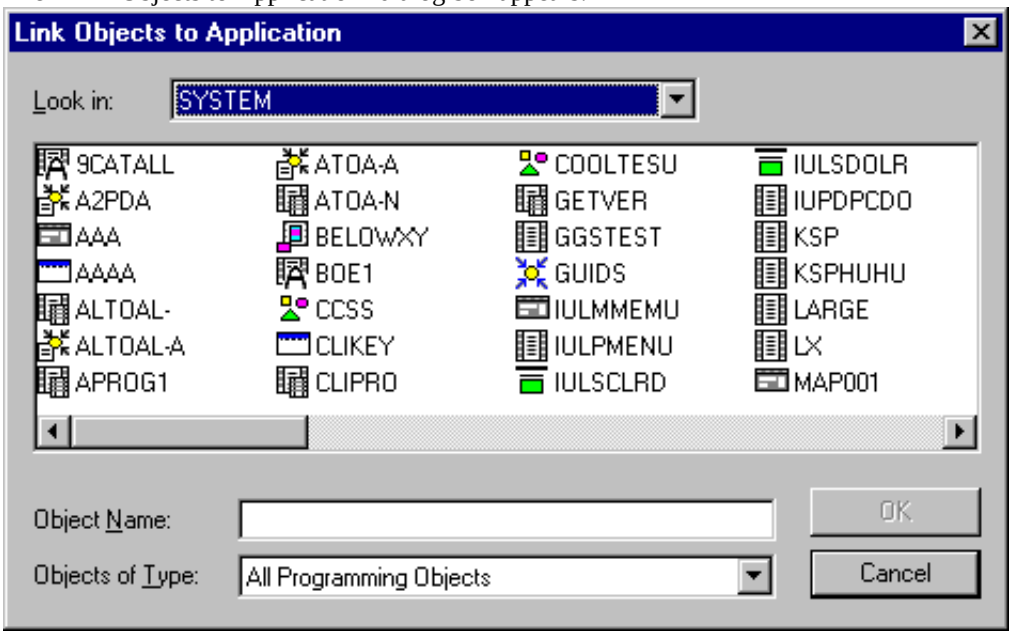
You can link any existing Natural object of the attached development server (for example, a program or map) to your application.

With the exercise below, you will link the following objects to your base applications:

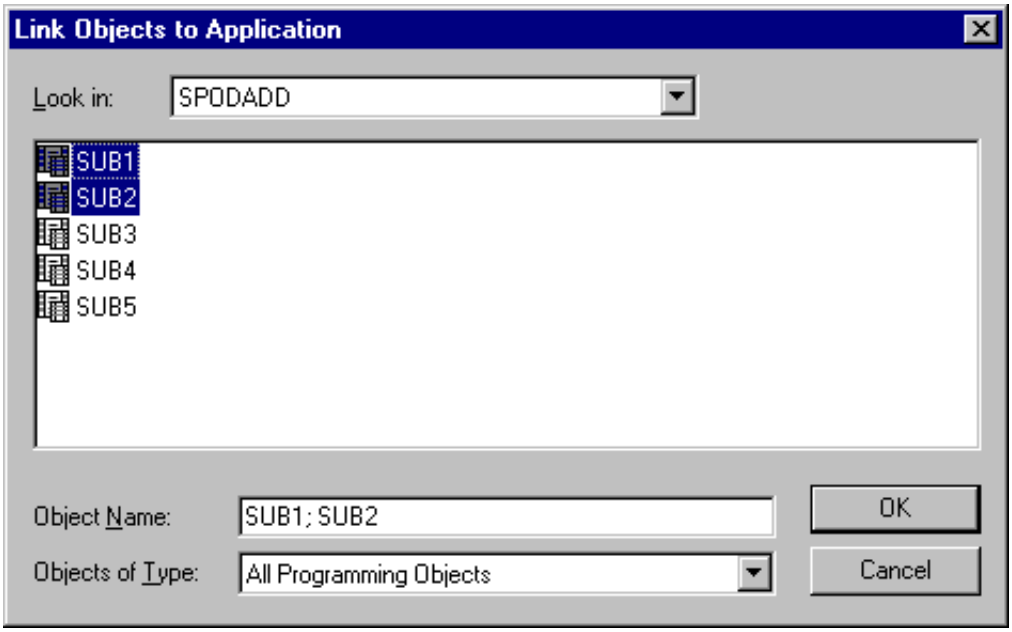
	SPODAPPL1	SPODAPPL2
SPODADD	SUB1 SUB2	SUB2
SPODLIB	MAINPGM MENU1 PROG3 TEXT1	MENU1 PROG1 TEXT2
SPODTEST		PGMCHECK

To link objects to a base application

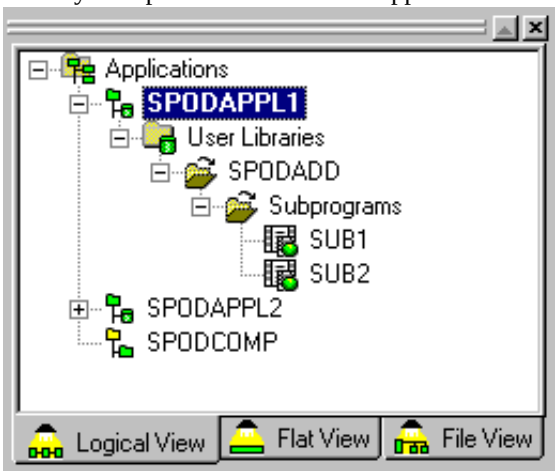
1. In the application workspace, click your base application SPODAPPL1 with the right mouse button.
2. From the resulting context menu, choose **Link**.
   The "Link Objects to Application" dialog box appears.



3. From the "Look in" drop-down list box, select the library SPODADD.
   The content of the selected library is now shown in the dialog box.
4. Select the objects SUB1 and SUB2.
   To select more than one object, press CTRL while selecting the objects with the mouse.



5. Choose the **OK** button.
   When you expand the nodes in the application workspace, the linked objects are shown.



   Note:
   In logical view, the selected objects are automatically placed in the corresponding folders (i.e. a "Programs" folder is shown for all programs that you have selected).
6. Repeat the above steps to link the remaining objects as indicated in the above table.

Note:
If you notice that you have linked a wrong object to your base application, you can unlink it. To do so, select the wrong object in the application workspace, click the right mouse button and from the resulting context menu, choose **Unlink**.

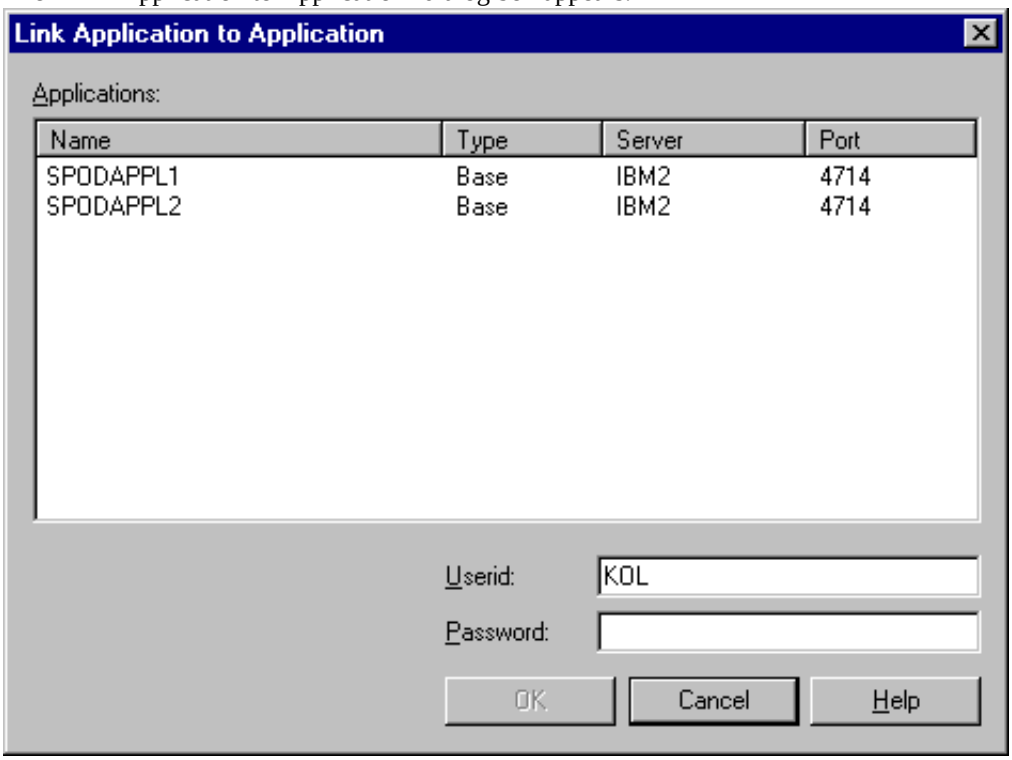
Linking Base Applications to a Compound Application

You can link any existing base application on the attached development server to a compound application. You can also link base applications that are stored on different servers. A compound application thus allows you to combine objects that cannot be combined in library workspace.

You will now link your two base applications SPODAPPL1 and SPODAPPL2 to your compound application SPODCOMP.

To link base applications to a compound application

1. In the application workspace, click your compound application SPODCOMP with the right mouse button.
2. From the resulting context menu, choose **Link**.
   The "Link Application to Application" dialog box appears.



3. Select your first base application named SPODAPPL1.
4. Optionally. If a password is required, specify it in the "Password" text box.
   If you do not specify a required password, an additional dialog box will later prompt you for this information.
5. Choose the **OK** button.
6. Repeat above steps to link second base application named SPODAPPL2.

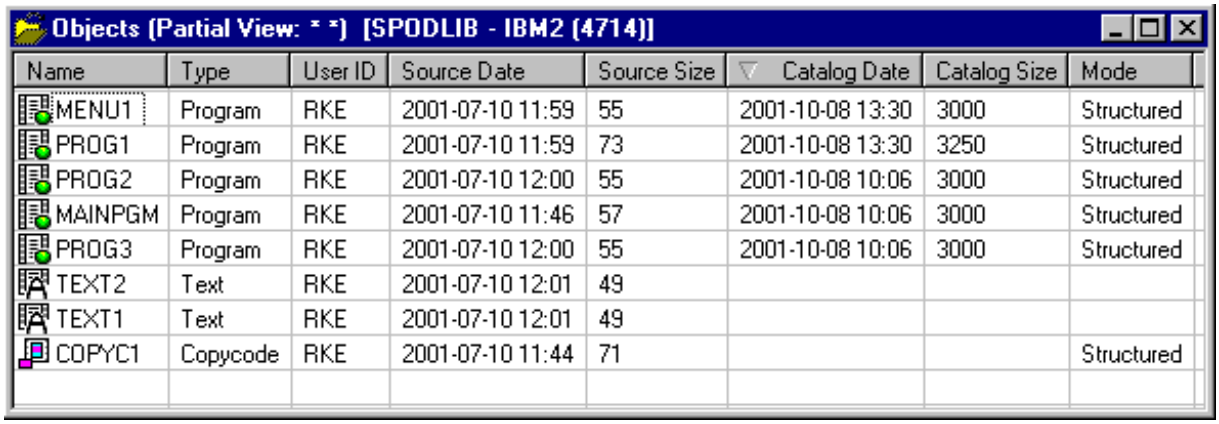
Managing Linked Objects

The objects in the application workspace are just references (or links) to the objects on the development server. They are not copies. For example, when you add a new program it will be visible in both the library workspace and the application workspace.

When you want to modify an object, you can do this either in the library workspace or in the application workspace. When an object is currently being modified by another user, the corresponding lock message is shown. A lock message is also shown when you try to open an object in library workspace that you are currently modifying in application workspace (and vice versa).

Note:
Not all commands are available in application workspace. For example, the commands **Delete** and **Rename** are only available in library workspace.

The following exception applies in application workspace when cataloging (CATALL) the objects in a library: only the objects that have been linked with the application are cataloged (i.e. only the objects that are shown in the application workspace). The objects that are only shown in the library workspace are ignored. To find out which objects have been cataloged for the library that is currently selected in library workspace, issue the `LIST *` command from the command line. You can then check the catalog date in the resulting window. The catalog date in the following example shows that only the first two objects have been cataloged recently.



Name	Type	User ID	Source Date	Source Size	Catalog Date	Catalog Size	Mode
MENU1	Program	RKE	2001-07-10 11:59	55	2001-10-08 13:30	3000	Structured
PROG1	Program	RKE	2001-07-10 11:59	73	2001-10-08 13:30	3250	Structured
PROG2	Program	RKE	2001-07-10 12:00	55	2001-10-08 10:06	3000	Structured
MAINPGM	Program	RKE	2001-07-10 11:46	57	2001-10-08 10:06	3000	Structured
PROG3	Program	RKE	2001-07-10 12:00	55	2001-10-08 10:06	3000	Structured
TEXT2	Text	RKE	2001-07-10 12:01	49			
TEXT1	Text	RKE	2001-07-10 12:01	49			
COPYC1	Copycode	RKE	2001-07-10 11:44	71			Structured

Mapping an Application

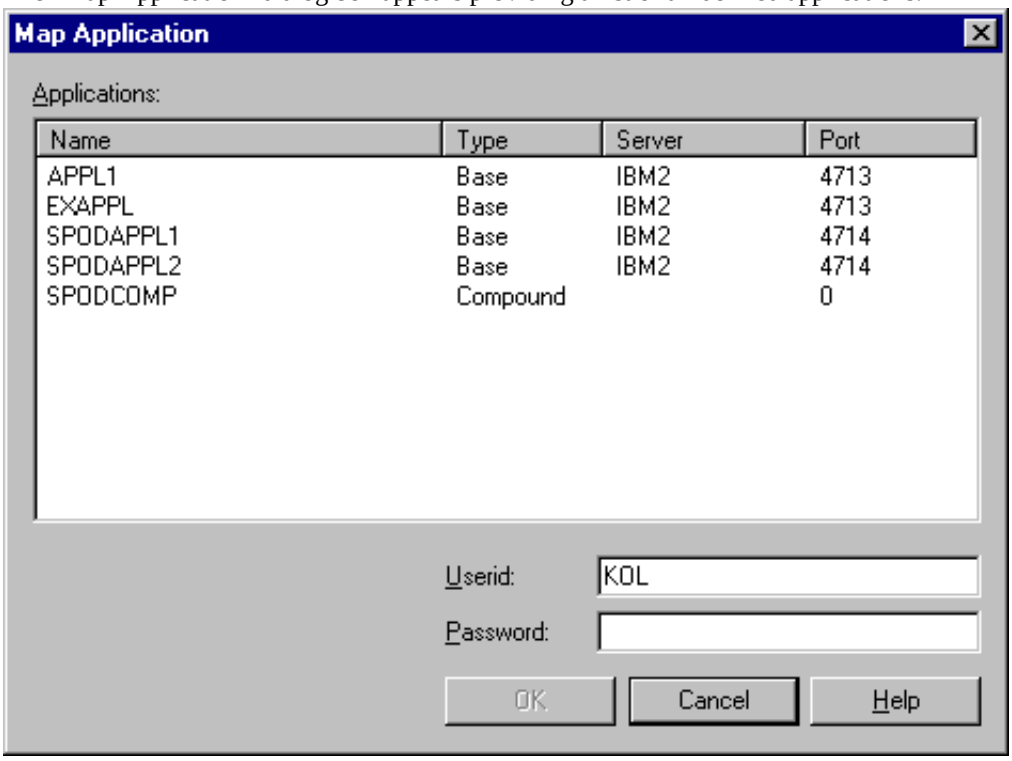
When you map an application, it is shown in the application workspace. You can map all applications that have already been defined on the development server that you have defined in the "Map Application Server" dialog box (i.e. the development server on which the Application Manager is located).

Each time you restart Natural Studio, your application workspace is empty. To display the previously mapped applications, click the plus sign next to the node name "Applications". As a result, all previously mapped applications are mapped again. If a password is required, further dialog boxes may appear in which you have to specify the missing information.

The following exercise explains how to map applications that you have not previously created (for example, applications that have been created by other users).

To map an application

1. In the application workspace, click the node name "Applications" with the right mouse button.
2. From the resulting context menu, choose **Map**.
   The "Map Application" dialog box appears providing a list of all defined applications.



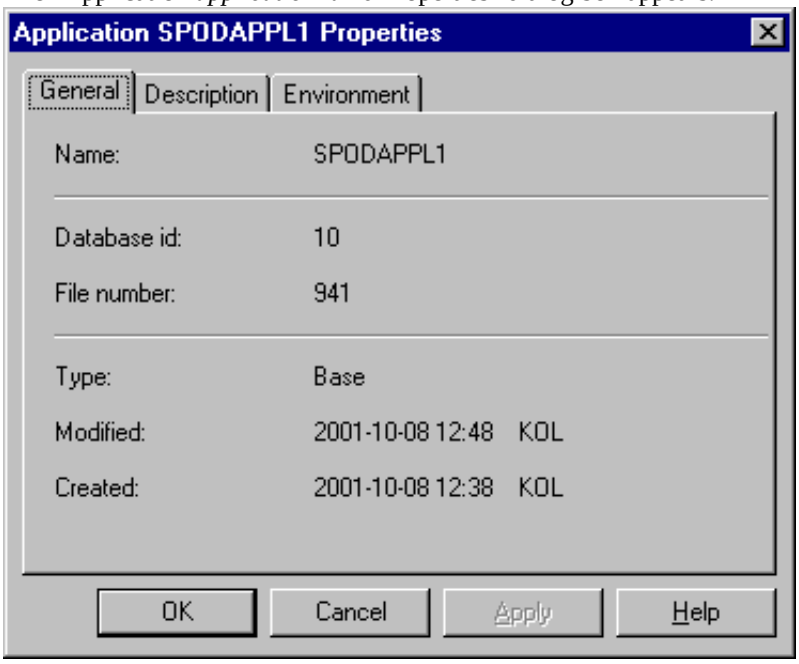
3. Select the application you want to map.
4. Optionally. If a password is required, specify it in the "Password" text box.
   If you do not specify a required password, an additional dialog box will later prompt you for this information.
5. Choose the **OK** button.
   The application is now shown in the application workspace.

Displaying the Properties of an Application

If you need information about an application in the application workspace or if you want to change the settings of the application, you can display its properties.

To display the properties of an application

1. In the application workspace, click the application for which you want to display the properties with the right mouse button.
2. From the resulting context menu, choose **Properties**.
   Or press ALT+ENTER.
   The "Application *applicationname* Properties" dialog box appears.



3. Click the tabs "Description" and "Environment" to view the corresponding properties.
4. Choose the **OK** button to close the dialog box.

You can now proceed with the next exercise: Displaying Cross-Reference Data

Displaying Cross-Reference Data

Using the XRef GUI Client plug-in, you can navigate through cross-reference data in a development server file. These cross-reference data are created when you catalog or stow an object. For detailed information, see Natural XRef GUI Client in the Remote Development documentation of Natural for Windows.

Generation of cross-reference data was previously only available with Predict. With the Natural development server and the Natural XRef GUI Client, it is now also available without Predict.

The following topics are covered below:

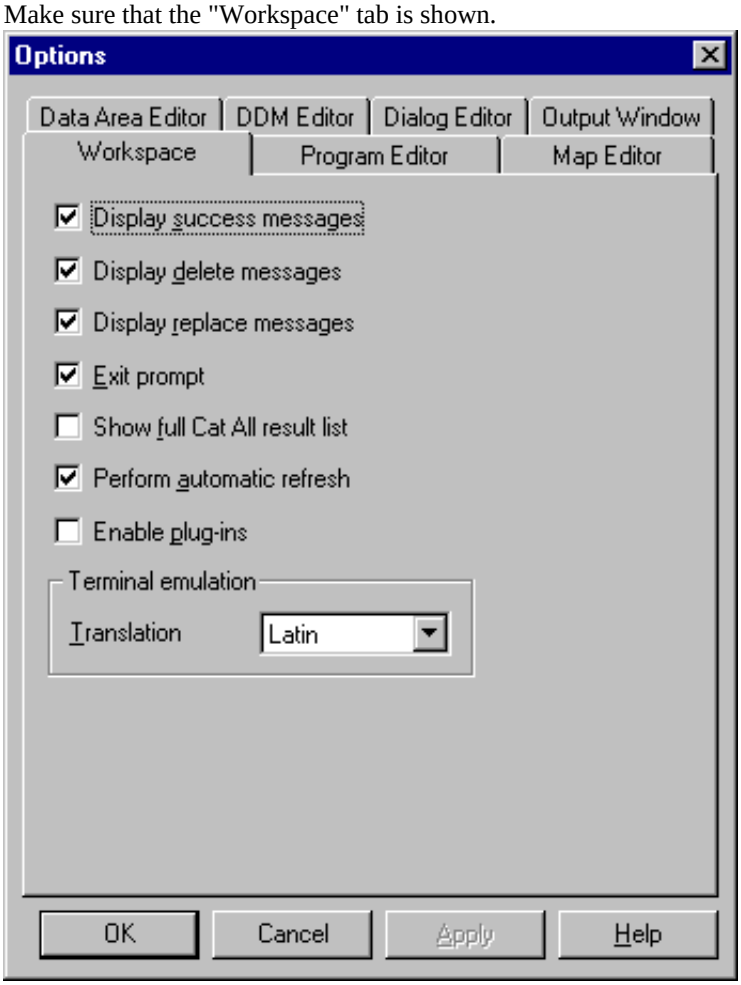
- Enabling the Usage of Plug-Ins
- Activating the XRef GUI Client Plug-In
- Activating the Generation of Cross-Reference Data
- Generating Cross-Reference Data
- Displaying Active Cross-References
- Applying a Change to a Copycode
- Displaying Passive Cross-References
- Cataloging the Objects that Include the Changed Copycode

Enabling the Usage of Plug-Ins

So that you can use the XRef GUI Client plug-in, you must first enable the usage of plug-ins. Once enabled, the Plug-in Manager is automatically active each time you start Natural.

To enable the usage of plug-ins

1. From the **Tools** menu, choose **Options**.
   The "Options" dialog box appears.
2. Make sure that the "Workspace" tab is shown.



   Note:
   If you have previously disabled success and delete messages, the corresponding check boxes are not selected.
3. Make sure that the "Enable plug-ins" check box is selected.
4. Choose the **OK** button.
   The **Plug-in Manager** command and the corresponding toolbar button are now available (see below). When the usage of plug-ins is not enabled, this command and toolbar button are not shown.

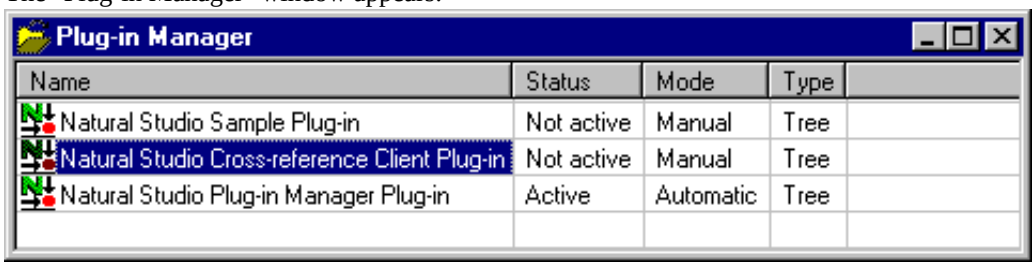
Activating the XRef GUI Client Plug-In

The XRef GUI Client plug-in is not active by default.

When the Plug-in Manager has been enabled, you can activate the XRef GUI Client plug-in. Two types of activation mode are available:

- **Automatic**
  The XRef GUI Client plug-in is automatically activated each time Natural is started.
- **Manual**
  The XRef GUI Client plug-in must be activated manually on demand (default).

To activate the XRef GUI Client plug-in

1. From the **Tools** menu, choose **Configuration Tools > Plug-in Manager**.
   Or click the following toolbar button:
   
   The "Plug-in Manager" window appears.
   

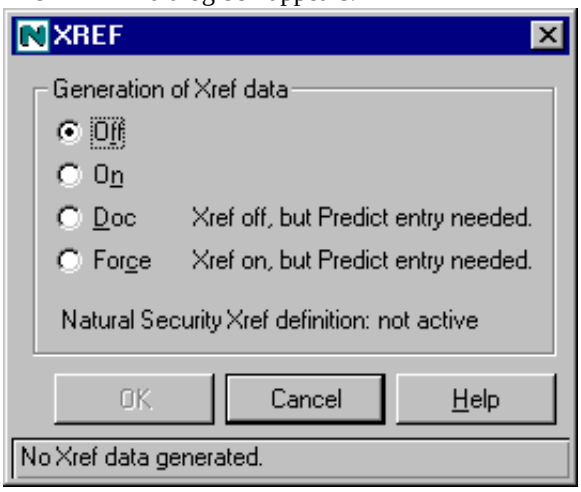
2. Select the "Natural Studio Cross-reference Client Plug-in" entry.
3. Click the right mouse button and from the resulting context menu, choose **Activation mode > Automatic**.
   This does not automatically activate the XRef GUI Client for the current session. To activate it, you can either restart Natural or proceed as described with the next step.
4. Click the right mouse button while "Natural Studio Cross-reference Client Plug-in" is still selected and from the resulting context menu, choose **Activate**.
   When activated, the XRef toolbar is automatically provided:
   
   Note:
   If you want to deactivate the XRef toolbar, choose **Customize** from the **Tools** menu. See Displaying Additional Toolbars for further information.

Activating the Generation of Cross-Reference Data

Whether the generation of cross-reference data is activated by default depends on the settings of the Natural parameter module. You will now make sure that cross-reference data will be generated.

To activate the generation of cross-reference data

1. From the **Tools** menu, choose **Configuration Tools > XRef Generation**.
   The "XREF" dialog box appears.



2. Make sure that the "On" option button is selected.
3. Choose the **OK** button.
   A message appears indicating that Xref mode is now set to On.
4. Choose the **OK** button.
   Cross-reference data will now be generated the next time you catalog an object.

Generating Cross-Reference Data

Cross-reference data are generated when cataloging objects.

For this tutorial, you will first copy all objects from the system library SYSSPODX to a new user library named SPODXREF. To generate cross-reference data, you will then catalog the objects in the new user library.

To copy the demo data to a new user library

1. Create a new user library named SPODXREF (see Creating User Libraries).
   Make sure to create this library in an environment in which a Natural development server has been installed.
2. Copy all objects from the system library SYSSPODX to the new library SPODXREF (see Copying and Moving Objects).

To generate cross-reference data

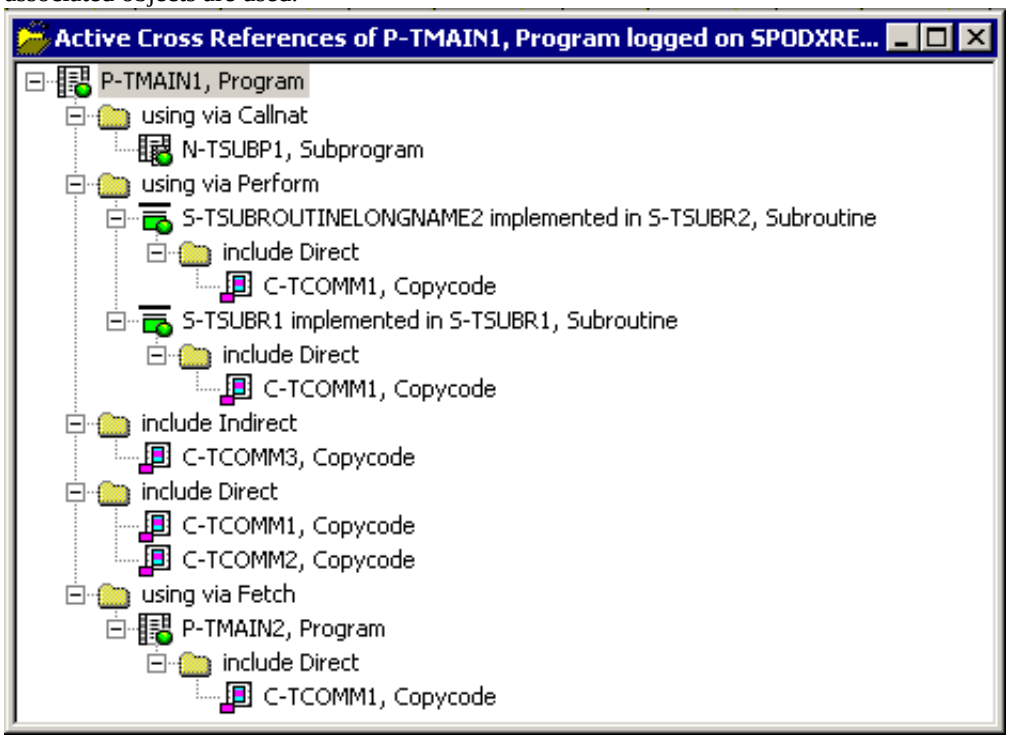
- Catalog all objects in the new library SPODXREF (see Cataloging Objects).
  Cross-reference data are now available for all objects in this library.

Displaying Active Cross-References

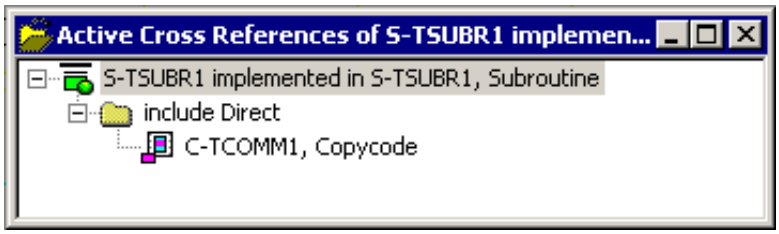
You will now check which objects are used by the program P-TMAIN1 in your new library SPODXREF.

To display the active cross-references for the program P-TMAIN1

1. Select the program P-TMAIN1 in the library SPODXREF.
2. From the **Tools** menu, choose **Development Tools > XRef GUI Client > Selected Objects > Active Cross References**.
   Or click the following toolbar button:
   
   Tip:
   Click the program with the right mouse button. You can then choose **Active Cross References** from the resulting context menu.
   A tree view window appears showing the active cross-references for the selected program.
3. Expand the nodes under P-TMAIN1 (by clicking the plus signs) to display the logical structure in which the associated objects are used.



Note:
Expanding the node, for example, for the subroutine S-TSUBR1 in the above example displays the same data as selecting this subroutine in the library workspace (for example, in logical view) and executing the command **Active Cross References**:

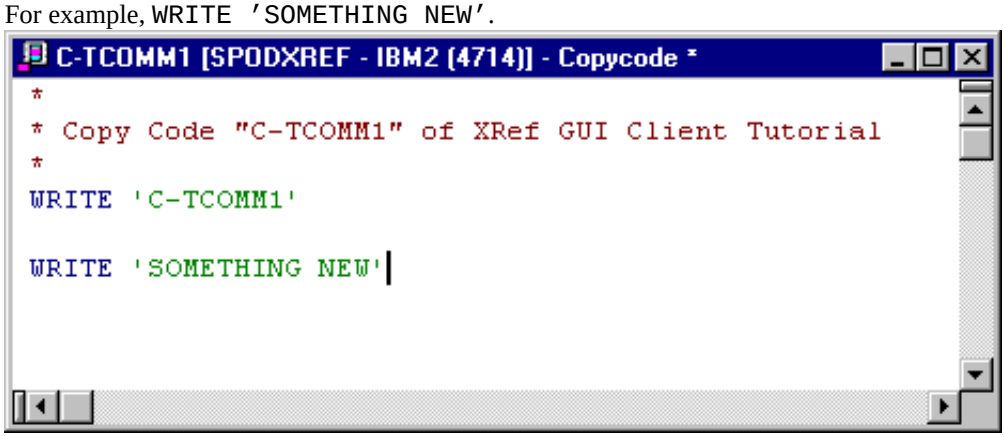


Applying a Change to a Copycode

You will now apply a small change to the copycode C-TCOMM1. You will later check by which objects this changed copycode is used so that you can catalog these objects once more.

To apply a change to a copycode

1. Click C-TCOMM1 with the right mouse button and from the resulting context menu, choose **Open**.
   Or double-click C-TCOMM1.
   This can be done either in the library workspace or in the tree view window showing the active cross-references. In the latter case, you can open the copycode by using the **Open** command with any node representing C-TCOMM1 (under the nodes "using via Fetch", "using via Perform" or "include Direct").
   The copycode is now shown in an editor window.
2. Add a new statement to the copycode.
   For example, `WRITE 'SOMETHING NEW'.`



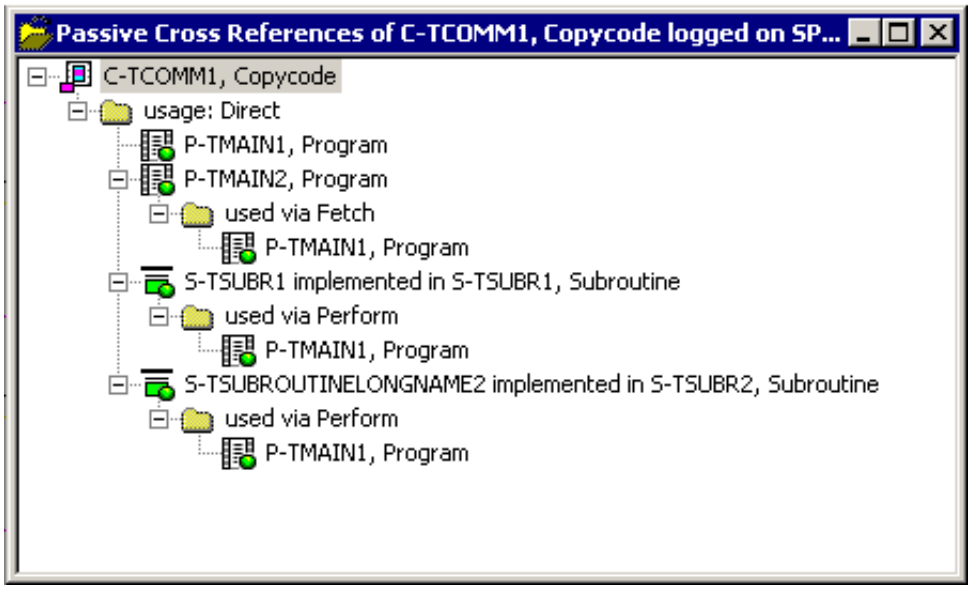
3. From the **Object** menu, choose **Save**.
   Or press CTRL+S.
   Or click the following toolbar button:
   
4. Click the standard close button at the top right of the editor window to close this window.

Displaying Passive Cross-References

You will now check by which objects the changed copycode is used.

To display the passive cross-references for the copycode C-TCOMM1

1. Select the copycode C-TCOMM1 in the tree view window showing the active cross-references.
2. From the **Tools** menu, choose **Development Tools > XRef GUI Client > Selected Objects > Passive Cross References**.
   Or click the following toolbar button:
   
   Tip:
   Click the copycode with the right mouse button. You can then choose **Passive Cross References** from the resulting context menu.
   A tree view window appears showing the passive cross-references for the selected copycode.
3. Expand the nodes under C-TCOMM1 to display the objects that include C-TCOMM1.



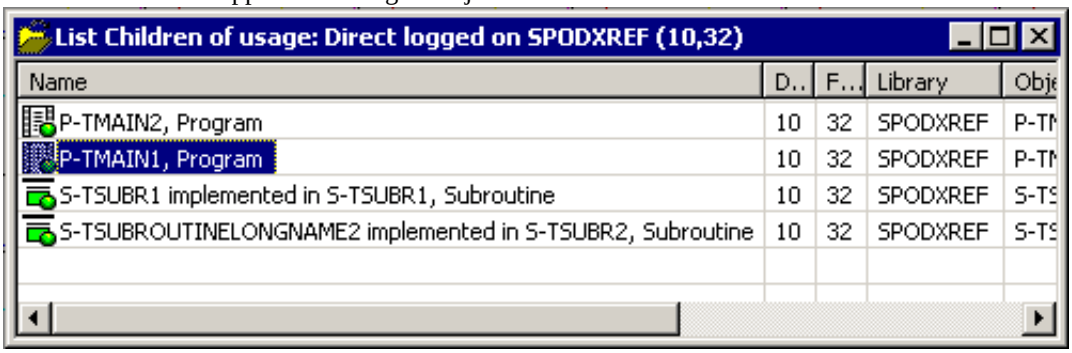
Cataloging the Objects that Include the Changed Copycode

You will now catalog the objects which include the changed copycode.

When you proceed as described below, you can catalog a number of objects at the same time. The tree view window showing the passive cross-references (see above) only allows you to select one object at a time.

To catalog the objects that include C-TCOMM1

1. In the tree view window showing the passive cross-references, click the node name "usage: Direct" with the right mouse button.
2. From the resulting context menu, choose **List Children**.
   A list view window appears showing all objects.



3. Press CTRL+A to select all objects in the list.
4. Click the right mouse button and from the resulting context menu, choose **Catalog**.
   Or click the following toolbar button:
   

You have now successfully completed this tutorial.

SPoD Administration Policies and Procedures - Overview

SPoD Administration Policies and Procedures - Overview

This documentation provides information which is required to be able to administer a Natural development environment that is based on the Natural Single Point of Development (SPoD) approach.

It is primarily aimed at those who are responsible for setting up an effective working environment for Natural application developers and covers the following topics:

Administration Policies and Procedures

- Installing Natural Single Point of Development
- Configuring Natural Single Point of Development
- System Management Hub Installation for Natural Manager
- Using the System Management Hub with Natural

Installing Natural Single Point of Development

The following installation guide summarizes the prerequisites and the installation steps that are necessary to create a Natural Single Point of Development environment that uses Natural Studio as a remote development client and the Natural Development Server plug-in as a remote development server for Mainframes running under the operating system OS/390.

The following topics are covered:

- Prerequisites
- Installation Procedure

Prerequisites

Products Required or Involved

The following products are involved:

- Natural Version 5.1.1 for Windows
- Natural Development Server Version 1.1.1 for OS/390
- Natural Version 3.1.5 or higher for Mainframes
- System Management Hub
- Predict Version 4.2 for Mainframes (optional, if you wish to continue using Predict)

Technical Requirements

If you are working in a wide-area network (WAN) environment, please note that the minimum transmission rate must be 10 Mbit or higher.

The speed of a modem or an ISDN connection will be insufficient.

Installation Procedure

Installing the products that make up Natural Single Point of Development comprises the following general installation steps:

- Installing Natural Studio
- Installing Natural for Mainframes
- Installing the Natural Development Server
- Installing the System Management Hub

Installing Natural Studio

- **General Installation**

  Refer to Installing Natural for Windows.

- **Installing the SPoD-Specific Features of Natural Studio**

  If you wish to have the SPoD-specific parts of Natural Studio enabled, select the setup type referring to server development (SPoD) in the course of Natural for Windows installation procedure.

Installing Natural for Mainframes

Refer to the relevant section of the Natural for Mainframes Installation Guide:

- Installing Natural under OS/390
- (other SPoD-enabled platforms in preparation)

For background information on the automated installation of Software AG products, see also Concepts of System Maintenance Aid in the System Maintenance Aid documentation.

Installing the Natural Development Server

Refer to the relevant section of the Natural Development Server installation instructions:

- Installing the Natural Development Server under OS/390
- (other SPoD-enabled platforms in preparation)

Defining a Natural Development Server File

The Natural Development Server File is used as a central dictionary file for storing Natural applications and the links to objects making up an application. It also holds object locking information. This information is not bound to certain groups of application developers, but has an impact on the entire application development of an enterprise. Therefore, this file should be available only once, to ensure that the application definitions and locking states are kept consistent.

To ensure the use of one single Natural development file

1. Set the NTDYNP parameter in the default Natural parameter module NATPARM to OFF:
   - Specify `NTDYNP OFF`, if you want to disallow overwriting off all dynamic parameters
   - or specify `NTDYNP OFF,FDIC`, if you want to disallow overwriting of the Natural development server file only.
2. Disallow any changes to profiles by the end user.

Installing the System Management Hub

Software AG's System Management Hub must have been installed as described in the document System Management Hub Installation for Natural Manager.

Configuring Natural Single Point of Development

This section provides an overview of the configuration facilities that exist for the individual products involved in a Natural Single Point of Development environment. References are given to documents where the relevant configuration functions or parameters are described.

The following topics are covered:

- Configuring Natural for Windows
- Configuring Natural for Mainframes
- Configuring the Natural Development Server

See also Operating the Natural Development Server.

Configuring Natural for Windows

Natural for Windows offers configuration possibilities which are documented in the Natural for Windows Operations documentation. For details, refer to:

- Local Configuration File
  - Buffer Pool Assignments
  - Installation Assignments
- Global Configuration File
  - DBMS Assignments
  - Dictionary Server Assignments
  - Printer Profiles
  - System File Assignments

Configuring Natural for Mainframes

Information on how to configure Natural in a mainframe environment is primarily contained in the Natural Operations for Mainframes and in the Natural Reference documentation. The following topics are relevant:

- Profile Parameter Usage
  - Natural Parameter Hierarchy
  - Assignment of Parameter Values
  - Profile Parameters Grouped by Function
- Profile Parameters

Configuring the Natural Development Server

For details on how to configure the Natural Development Server, refer to:

- Development Server Configuration under OS/390

System Management Hub Installation for Natural Manager

For Natural Manager, the System Management Hub server is installed on the mainframe under OS/390. The client is a Javascript-enabled browser (Netscape 4.7 or Internet Explorer 4.01, 5.0, and 5.5).

This document is organized as follows:

Before You Install

The installation of Natural Manager includes the installation of a server environment with System Management Hub (ARG), Software AG base technology (BTY), and a runtime system (APS).

Under OS/390, the server environment runs in a single address space. Within that address space, the System Management Hub servers (HTTP, MIL, and agent) run using the runtime system.

Before you start the installation procedure, you must have the following environment available:

- OS/390 version 2.6 or above;
- For RACF or ACF2 security privileges, the server environment started task (SSE) must have a USS segment (formerly called an OE segment) for a socket;
- Access to the IBM TCP/IP stack: this means the user IDs must be USS users;
- 150 cylinders of space on 3390 disks must be available for the installation datasets;
- Three (3) free port numbers for the HTTP, MIL, and agent servers.

Contents of the Installation Tape

You will receive a tape containing the mainframe files. There are two components on the tape:

- Natural Development Server
- System Management Hub for Natural

The following datasets refer to the System Management Hub for Natural. For details on the Natural Development Server part, refer to Natural Development Server Installation under OS/390.

Tape Dataset	Dataset Type	Description
NDV112.AZIP	Binary data, PS, (U, 6447)	Data to be loaded into the portable file system (PFS) including the XML templates for Natural Manager
NDV112.JOBS	PO, (FB, 6320, 80), EBCDIC	Jobs for installing, administering, and operating Natural Manager
NDV112.LB00	Load library, PO, (U, 6447)	Batch executables
NDV112.LE00	Load library, PO, (U, 6447)	Server environment agents for Natural Manager
NDV112.SARG	PO, (FB, 10240, 512), EBCDIC	Registry data for the runtime system
APS251.SXML	PO, (FB, 10240, 512), EBCDIC	System Management Hub templates for the runtime system
APS251.LB00	Load library, PO, (U, 6447)	Batch environment agents for the runtime system
APS251.LD00	Load library, PO, (U, 6447)	Load library for the runtime system
APS251.LE00	Load library, PO, (U, 6447)	Server environment agents for the runtime system
APS251.SRCE	PO, (FB, 6320, 80), EBCDIC	Source and jobs dataset for the runtime system
ARG122.AZIP	Binary data, PS, (U, 6447)	Zip file for System Management Hub
ARG122.SARG	PO, (FB, 10240, 512), EBCDIC	Registry data for System Management Hub
ARG122.LB00	Load library, PO, (U, 6447)	Batch environment agents for System Management Hub
ARG122.LD00	Load library, PO, (U, 6447)	Load library for System Management Hub
ARG122.LE00	Load library, PO, (U, 6447)	Server environment agents for System Management Hub
ARG122.SRCE	PO, (FB, 6320, 80), EBCDIC	Source and jobs dataset for System Management Hub
BTY115.ECSO	Binary data library, PO, (U, 4096)	Translation tables for Software AG base technology
BTY115.LB00	Load library, PO, (U, 6447)	Batch environment agents for Software AG base technology
BTY115.LD00	Load library, PO, (U, 6447)	Load library for Software AG base technology

Contents of the Jobs Library

The NDVvrs.JOBS library contains

- Jobs for installation steps (J#cccccc);
- JCL procedures and sample jobs used in installation (P#cccccc);
- Configuration parameter members (cccccccc); and
- Other sample jobs (X#cccccc).

Note:
Configuration members are copied from the SARG dataset to the JOBS dataset during step 2 when applying parameter replacements.

Jobs for Installation Steps

JCL Member	Purpose
J#COPYM	Installation step 1: Allocates and populates user-modifiable JCL libraries
J#PARSUB	Installation step 2: Sets configuration parameters in jobs, procedures, and parameter members
J#PFSINI	Installation step 3: Allocates and formats the portable file system (PFS) containers
J#REGINI	Installation step 4: Creates and loads the registry into the file systems.
J#UNZIP	Installation step 5: Loads the System Management Hub and the runtime system data

JCL Procedures and Sample Jobs Used in Installation

JCL Member	Description
P#ARGB	ARGBATCH procedure
P#BAT	Batch procedure
P#PARSUB	Procedure for the substitution job J#PARSUB
P#SRV	Server procedure (started task)

Configuration Parameter Members

Parm Members	Description
ENV	Server environment variables
ENVB	ARGBATCH environment variables
PARSUB	Current values for the J#PARSUB job
PARSUBEG	Example of current values for the J#PARSUB job
REGCONV	Registry configuration
SYPARGB	ARGBATCH sysparm member
SYPBAT	Batch sysparm member
SYPINIT	Initial sysparm member
SYPSRV	Server sysparm member

Other Sample Jobs

JCL Members	Purpose
X#PFSDIS	Job to display PFS contents
X#PFSEXP	Job to export files from PFS
X#PFSIMP	Job to import files to PFS
X#REGUPD	Job to update registry options
X#UNZIP	Job to unzip into PFS
Z#UNZIP	Copyright disclaimer for UNZIP
X#REGNUC	Job to register Natural nuclei
X#REGSYS	Job to register Natural system files

Copying the Installation Tape to Disk

If you have Software AG's System Management Aid (SMA) installed, you may use it to copy the Natural Manager installation tape to disk. For information about using SMA, refer to the System Maintenance Aid Manual.

Note:
SMA does not support the entire installation process; only copying the tape to disk.

Copying the Tape Contents to Disk

If you are not using SMA: Copy the job dataset NDV*nnn*.JOBS from tape to disk using the sample JCL below.

The following values must be supplied in the JCL:

1. In the dataset names, replace *nnn* with the current version number of the datasets.
2. With the SER parameter, replace *XXXXXX* with the volume serial number of the tape .
3. With the LABEL parameter, replace *x* with the sequential number of the tape dataset (see the **Report of Tape Creation**).
4. With the VOL=SER parameter, replace *YYYYYY* with the volume serial number of the disk pack.

```
// JOB CARD
//V2COPY   EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=A
//IN1      DD DSN=NDVnnn.JOBS,DISP=OLD,UNIT=TAPE,
//   VOL=(,RETAIN,SER=XXXXXX),LABEL=(x,SL)
//OUT1     DD DSN=SAGLIB.NDVnnn.JOBS,DISP=(NEW,CATLG,DELETE),
//   UNIT=SYSDA,VOL=SER=YYYYYY,SPACE=(CYL,(1,1,10))
//SYSIN    DD *
C I=IN1,O=OUT1 /*
```

Adapt and run job NDVTAPE from the job dataset to copy the load and source libraries from tape to disk, unless already done during the installation of the Natural Development Server part.
Note that the job NDVTAPE contains the JCL to copy the datasets of both the Natural Development Server and the System Management Hub for Natural.
The sample jobs directly use the sequential datasets from tape

Installation Overview

After unloading the installation tape to disk, install Natural Manager using the following steps:

1. Allocate and populate user-modifiable copies of the JCL files (job J#COPYM).
2. Set the configuration parameters in jobs, procedures, and parameter members (job J#PARSUB).
3. Allocate and format the PFS files (job J#PFSINI).
4. Set up the registry files (job J#REGINI).
5. Initialize the System Management Hub (job J#UNZIP).
6. (Optional) Update the registry options (job X#REGUPD).
7. Register nuclei and system files.
8. Start the JCL server procedure P#SRV as a started task.

Rerunning Installation Steps

You can rerun installation steps when a parameter changes, for example. Note the following, however:

- If you change parameters in step 2, you must repeat all following steps except step 7.
- If you repeat step 4 or 5, you may receive errors related to changes (such as the creation of a subdirectory) that were made to the portable file system (PFS) in the previous execution.

Condition Codes

The System Management Hub utility ARGEXEC in J#PFSINI, J#REGINI, and J#UNZIP behaves like a shell in which programs execute. An error from a program running in the shell is not passed as a condition code, which means that you need to check the job output to determine successful completion.

Step 1. Allocate and Populate Copies of the JCL Libraries

Member J#COPYM in the NDV112.JOBS dataset provides JCL to allocate and populate user-modifiable copies of the JCL libraries.

To allocate and populate copies of the JCL libraries

1. Adapt the job J#COPYM in the NDV112.JOBS dataset to your site requirements. Set a value for the job statement; supply the SET statements with the same values you use for the PARSUB member (see step 2).
2. Submit J#COPYM.

Step 2. Set the Configuration Parameters

Note:
The PARSUB configuration variables provide a basic setup for the product. Once you have this setup, you can modify other variables in other members for a more customized configuration, as required.

This step comprises the following substeps:

1. Set configuration parameters in the PARSUB member in the NDV112.JOBS dataset.
2. Adapt the job J#PARSUB in the NDV112.JOBS dataset to your site requirements.
3. Run the J#PARSUB job.

Step 2.1 Configure the PARSUB Member

The variables in the PARSUB member can be replaced. Ensure that the replacement value is after the equals sign (=) and is enclosed in (double) quotation marks.

```
"<NATURAL_LOAD>"="Natural LOAD LIBRARY"
"<ADABAS_SVC>"="249"
"<HILEV>"="highlevel-dataset-qualifier-with-dot"
"<HILEV_SLASH>"="highlevel-dataset-qualifier-with-slash"
"<HOST>"="host-system-name"
"<JOBPAR>"="job-parameters-in-job-statement"
"<LOCATION>"="UNIT=xxxx,VOL=SER=vvvvvv"
"<MILTCPIP_HOSTNAME>"="internet-host-name"
"<MILHTTP_PORT>"="9991"
"<MIL_PORT>"="9990"
"<MYHIL>"="highlevel-dataset-qualifier-for-user-modifiable-jcl-datasets"
"<SC_PORT>"="9900"
"<SYSTCPD>"="systcpd-dataset-name"
"<TCP_ADDRESS_SPACE>"="jobname-of-tcpip-task"
"<USER_ID>"="userid"
```

The variables in the PARSUB member are described in the following table:

Variable	Description
NATURAL_LOAD	the Natural batch library. The dataset which contains the executable Natural nucleus.
ADABAS_SVC	the Natural SVC of the databases that Natural Manager will access (default 249).
HILEV	the high-level qualifier under which the datasets were unloaded from the installation tape using a period as a delimiter. These datasets are read-only.
HILEV_SLASH	the high-level qualifier of the database datasets using a forward slash (/) as a delimiter. These datasets are customized for the installation and are read/write (JOBS, SARG, PFS, PFSB).
HOST	the SSE system name used as identification in console messages.
JOBPAR	the parameters used in the JOB statement of Natural Manager installation jobs.
LOCATION	a valid storage location for the PFS container files.
MILTCPIP_HOSTNAME	the internet name of the host where the MIL server is to run; that is, the host-name - domain-name or the TCP/IP address. For standard installations, "localhost" is appropriate.
MILHTTP_PORT	an available IP port for the HTTP listener of the MIL server on the runtime system.
MIL_PORT	an available IP port for the MIL server on the runtime system.
MYHIL	the high-level qualifier under which job J#COPYM allocates user-modifiable copies of the JCL libraries using a period as a delimiter. These datasets are read/write.
SC_PORT	an available IP port for the System Management Hub daemon (agent server).
SYSTCPD	the TCPIP.DATA configuration file; that is, the dataset/member for the SYSTCPD DD statement of the IBM TCP/IP stack.
TCP_ADDRESS_SPACE	the job name of the TCP/IP stack that the server address space will use.
USER_ID	a valid user name for the person who will administer the System Management Hub environment. This is used as the job name prefix in the installation jobs. Additional names may be added later. **Note:** User names are case-sensitive. Software AG recommends entering the user name always in uppercase.

Note:
You are required to specify three IP ports during installation: <MILHTTP_PORT>, <MIL_PORT>, and <SC_PORT>, the mimumum number required for use by the System Management Hub.

An example of the PARSUB member with values replaced is provided in member PARSUBEG:

```
"<NATURAL_LOAD>"="AST.NAT315.LOAD"
"<ADABAS_SVC>"="235"
"<HILEV>"="AST.COMN.A2"
"<HILEV_SLASH>"="AST/COMN/A2"
"<HOST>"="AST3F"
"<JOBPAR>"="AST,CLASS=K,MSGCLASS=X"
"<LOCATION>"="UNIT=3390,VOL=SER=NAT005"
"<MILTCPIP_HOSTNAME>"="LOCALHOST"
"<MILHTTP_PORT>"="7335"
"<MIL_PORT>"="7336"
"<MYHIL>"="AST.COMN.A4"
```



```
"<SC_PORT>"="7337"
"<SYSTCPD>"="SYSM.DAEF.PARMLIB(TCPDAT3F)"
"<TCP_ADDRESS_SPACE>"="DA3FTCP"
"<USER_ID>"="AST"
```

Configuring Communications for the TCP/IP Stack

System Management Hub supports the IBM OpenEdition TCP/IP stack. In the PARSUB variables, you have provided

- access to the TCPIP.DATA configuration file by allocating it statically in the SYSTCPD DD statement in the runtime JCL. However, there are other ways to make this configuration file available to System Management Hub. For more information, refer to the appropriate IP configuration manual for your OS/390 and TCP/IP release.
- the job name and host name of the TCP/IP stack that the server address space will use.

Configuring the Environment

Certain configuration information is obtained from environment variables. The NDV112.JOBS dataset contains two sample configuration members: ENV and ENVB.

Accept the settings in this file as delivered with the following exceptions:

Time Zone Setting

Set the time zone in the TZ parameter for the local time. For example, the time zone setting for German daylight saving time is `TZ=CET-2`.

ENVB Values

Env. Variable	Appropriate Values	Explanation
XTSDSURL	tcpip://<XTS_HOST>:12731	Identifies the XTS directory server by host name or IP address. **Note:** Required only if you plan to run any XTS-enabled products in the server address space (SSE).

Step 2.2 Adapt the Provided J#PARSUB Job

Adapt the J#PARSUB member to your local standards and dataset names.

To adapt the provided JCL

1. Add a valid job card.
2. Replace all occurrences of the string `<HILEV>` with the high-level qualifier under which the datasets were unloaded from the installation tape using a period as a delimiter.

   For example, if the load library NDV112.LB00 was downloaded as XZY.NDV112.LB00, then `<HILEV>=XYZ`. If NDV112.LB00 was downloaded as XZY.A4711.NDV112.LB00, then `<HILEV>=XYZ.A4711`

3. Replace all occurrences of the string <HILEV_SLASH> with the high-level qualifier of the database datasets, but using a forward slash (/) as a delimiter instead of a period.
4. Replace all occurrences of the string <MYHIL> with the high-level qualifier under which job J#COPYM allocated and populated the user-modifiable copies of the JCL libraries.

Step 2.3 Run the J#PARSUB Job

This job copies the installation members from NDV112.SARG (input for the registry update) into the NDV112.JOBS dataset and applies the parameter substitutions defined in PARSUB.

Step 3. Allocate and Format PFS Files

Member J#PFSINI in the <MYHIL>NDV112.JOBS dataset provides JCL to allocate and format PFS files necessary for System Management Hub data that will contain the registry and System Management Hub files.

To allocate and format the PFS container files

- Submit J#PFSINI.

Step 4. Set up the Registry

The registry is used by a number of components in the Natural Manager and System Management Hub environments.

You have already put all relevant variables into the PARSUB member and the job J#PARSUB has created a job that you can submit to set up the registry.

Member REGCONV from the NDV*vrs*.JOBS dataset contains variable substitution values used in loading the registry for establishing System Management Hub on the OS/390 host. The required substitution values were made in the REGCONV member when you submitted J#PARSUB. No other modifications are required.

Warning:
The following section is provided for your information only. Do not modify REGCONV unless requested to do so by Software AG.

Once your registry is established, the values you provide are not easily changed. They can be changed, however.

Registry Template Substitution Values

Site-specific values (right side of assignment) are provided for the variables (left side of assignment) in the REGCONV member. The replaceable values are enclosed in angles (<>).

Note:
If you are requested to modify the values, use an editor to search for and replace all occurrences, as some are repeated multiple times.

Following is a copy of the NDV112.JOBS(REGCONV) member, a template for registry conversion values that governs how System Management Hub is established in the runtime system environment:

```
REGCONV member-------------------------------------------------
#
# Substitution values used to build the initial registry.
#
# This file is used to make value substitutions within the registry
#
```



```
# Note: The only values that should be changed are those provided
#   within angle brackets (<>). Specifically, there are several file
#   names and prefixes that may not be changed. Activities outside
#   the bounds of these sanctioned changes will result in a broken
#   system.
#
#-------------------------------------------------------------------------
* Natural Manager substitution values
#-------------------------------------------------------------------------
nma_exe_path=FILE://<HILEV_SLASH>/NDV112/LE00
nma_lib_path=FILE://<HILEV_SLASH>/NDV112/LE00
nma_ver.rel.sm=1.1.2
nma_verrelsm=112
nma_template_path=//files/NDV112/files/xml
*
#-------------------------------------------------------------------------
* System Management Hub substitutions
#-------------------------------------------------------------------------
* 1. Port Numbers:
*
* MILayer Service:
MIL_PORT=<MIL_PORT>
MILHTTP_PORT=<MILHTTP_PORT>
MIL_HOST=<LOCAL_HOST_NAME>
*
* CSLayer Service:
SC_PORT=<SC_PORT>
*
*
PFS_ARGDIR=/files/Argus
ARGDIR=FILE://<HILEV_SLASH>/BTY115
ARGVERS=V122
SAG_USER=<USER_NAME>
ECSOBJDIR=FILE://<HILEV_SLASH>/BTY115/ECSO
#-------------------------------------------------------------------------
* Runtime System substitution values
#-------------------------------------------------------------------------
user_name=<USER_NAME>
aps_inst_path=FILE://<HILEV_SLASH>/APS251
aps_exe_path=FILE://<HILEV_SLASH>/APS251/LE00
aps_lib_path=FILE://<HILEV_SLASH>/APS251/LD00
aps_template_path=/files/APS
#
```

Following are the registry template substitution values used for System Management Hub:

Variable	=Value	Comments
nma_exe_path	FILE://<HILEV_SLASH>/NDV112/LE00	The load library into which the Natural Manager (NDV) executables were installed.
nma_lib_path	FILE://<HILEV_SLASH>/NDV112/LE00	The load library into which the Natural Manager (NDV) executables were installed.
nma_ver.rel.sm	1.1.2	Natural Manager version, revision, and system maintenance level numbers separated by ’.’
nma_verrelsm	112	Natural Manager version, revision, and system maintenance level numbers concatenated.

Variable	=Value	Comments
nma_template_path	//files/NDV112/files/xml	The name of the subdirectory where XML templates were installed using the J#UNZIP job. This default value is assumed by the installation and should not be changed.
MIL_PORT	<MIL_PORT>	An available IP port for the MIL server on the runtime system.
MILHTTP_PORT	<MILHTTP_PORT>	An available IP port for the HTTP listener of the MIL server on the runtime system.
MIL_HOST	<LOCAL_HOST_NAME>	The host name where MIL server is to run; that is, the local host.
SC_PORT	<SC_PORT>	An available IP port.
PFS_ARGDIR	/files/Argus	A valid portable file system (PFS) directory path. This value is used when System Management Hub XML templates and other files (HTML, js, gif, etc.) are stored in the PFS. This is required when the intention is to run the MIL server on the runtime system.
ARGDIR	FILE://<HILEV_SLASH>/BTY115	Used in the dataset name prefix where XML templates are found. It then appends `FILES/XML/AGENTS` and `FILES/XML/AGENTS/BATCH` to this name to find required members.
ARGVERS	V122	Do not change this value.
SAG_USER	<USER_NAME>	A valid user name for the person who will administer the System Management Hub environment. Additional names may be added later. **Note:** User names are case-sensitive. Software AG recommends entering the user name always in uppercase.
ECSOBJDIR	FILE://<HILEV_SLASH>/BTY115/ECSO	The dataset containing translation objects.
user_name	<USER_NAME>	The user name of the person who will administer the runtime system nodes of the local host. This name will be validated. **Note:** User names are case-sensitive. Software AG recommends entering the user name always in uppercase.
aps_inst_path	FILE://<HILEV_SLASH>/APS251	The directory into which the runtime system (APS) load library was installed. The /APS241 part of the file name is the version level of the runtime system.
aps_exe_path	FILE://<HILEV_SLASH>/APS251/LE00	The load library into which the runtime system (APS) executables were installed.

Variable	=Value	Comments
aps_lib_path	FILE://<HILEV_SLASH>/APS251/LD00	The load library into which the runtime system (APS) executables were installed.
aps_template_path	/files/APS	The name of the file where XML templates were installed using the J#UNZIP job. This default value is assumed by the installation and should not be changed.

Note:
In the System Management Hub on OS/390, two types of files and file names are used:

- native OS/390 files; and
- files specified within the portable files system (PFS) in a UNIX-like manner.

It is important to be aware of the differences, and the context of the files and file names when you use them. For more information, see File Naming Specification for OS/390.

Establish the Registry

To format the registry file and process the template substitution values

1. Edit the job J#REGINI in the NDV112.JOBS dataset to meet the requirements at your site.
2. Submit the job J#REGINI.

In addition to processing the template substitution values, the job initializes two registry files: one for the server environment and one for the administrator's batch environment.

Step 5. Initialize the System Management Hub

Member J#UNZIP in the NDV112.JOBS dataset provides JCL for loading files required for the MIL server environment into the PFS container file that maps to the directory structure of PFS_ARGDIR when you initialize your registry, which is assumed to be `/files/Argus`.

To initialize System Management Hub

1. Edit the JCL member J#UNZIP to meet the requirements at your site.
2. Submit J#UNZIP to initialize System Management Hub.

The job creates the directory structures to contain the MIL files and imports the files from the installation zip file to the PFS.

Step 6. (Optional) Update the Registry Options

To update your registry options

1. Edit the job X#REGUPD to meet your site requirements.
2. Submit X#REGUPD.

Step 7. (Optional) Register Additional Nucleus and System Files

Step 7.1 Register Additional Nucleus

Member X#REGNUC in the <MYHIL>.NDV112.JOBS dataset provides JCL to register a Natural nucleus.

```
//<USER_ID>71 JOB <JOBPAR>
//*
//* -------------------------------------------------------------
//*
//* Execute agent to register Natural Nucleus.
//*
//* -------------------------------------------------------------
//*
//* Installation step 7:
//* Job can also be executed later, to register additional
//* Natural nucleus.
//*
//* Be sure to replace all occurrences.
//*
// SET HILEV=<HILEV>
// SET MYHIL=<MYHIL>
//*
// JCLLIB ORDER=&MYHIL..NDV112.JOBS
//*
//* -------------------------------------------------------------
//* register nucleus for server pfs
//* -------------------------------------------------------------
//REGNUC EXEC P#BAT
//PARMS DD *
natvers
Command=Register
Path=<NATURAL_LOAD>
Name=*
//*
//* -------------------------------------------------------------
//* register nucleus for batch pfs
//* -------------------------------------------------------------
//REGNUCB EXEC P#ARGB
//PARMS DD *
natvers
Command=Register
Path=<NATURAL_LOAD>
Name=*
/*
```

Modify the path(name of the dataset), where the natural nucleus is located and modify the name of the nucleus. If a '*' is used instead of a name, all nuclii in the dataset are registered.

Submit job X#REGNUC.

Step 7.2 Register System Files

Member X#REGSYS in the <MYHIL>.NDV112.JOBS dataset provides JCL to register Natural system files.


```
//<USER_ID>72 JOB <JOBPAR>
//*
//* -------------------------------------------------------------
//*
//* Execute agent to register Natural System File.
//*
//* -------------------------------------------------------------
//*
//* Installation step 7:
//* Job can also be executed later, to register additional
//* Natural system files.
//*
//* Be sure to replace all occurrences.
//*
// SET HILEV=<HILEV>
// SET MYHIL=<MYHIL>
//*
// JCLLIB ORDER=&MYHIL..NDV112.JOBS
//*
//* -------------------------------------------------------------
//* register system files for server pfs
//* -------------------------------------------------------------
//REGSYSS EXEC P#BAT
//PARMS DD *
natvers8
Command=Register
Database=<DATABASE_ID>
File=<FILE_NUMBER>
Password=
;
natvers8
Command=Register
Database=<DATABASE_ID>
File=<FILE_NUMBER>
Password=
;
//*
//* -------------------------------------------------------------
//* register system files for batch pfs
//* -------------------------------------------------------------
//REGSYSB EXEC P#ARGB
//PARMS DD *
natvers8
Command=Register
Database=<DATABASE_ID>
File=<FILE_NUMBER>
Password=
;
natvers8
Command=Register
Database=<DATABASE_ID>
File=<FILE_NUMBER>
Password=
;
/*
```

Replace <DATBASE_ID> by the number of the system files database and <FILE_NUMBER> by the file number of the system file. Enter the password, if the system file is protected.

Submit job X#REGSYS.

Step 8. Start the System Management Hub JCL Procedure

The JCL procedure P#SRV in the dataset NDV112.JOBS contains JCL to execute System Management Hub servers on the mainframe.

To start the JCL procedure

1. Edit the P#SRV member of the NDV112.JOBS dataset to meet the requirements at your site.

   Note:
   TCPIP.DATA must be available either by using the system default search (`TCPIP.TCIPI.DATA` or `SYS1.TCP.DATA`) or by using the `SYSTCPD DD` card in the JCL.

   The following DD statements have uses as indicated:

DD Name	Description
STDOUT	Used for miscellaneous output.
STDERR	Used for miscellaneous error and trace output.
APSLOG	Processes messages from the runtime system environment.
SYSPARM	Contains the runtime parameter input definitions.
CONFIG	Contains environment variable definitions.
PFS	Portable file system (PFS) container files.
SYSTCPD	Configuration file for the IBM OE TCP/IP stack.

2. Copy the member into a system procedure library (proclib).
3. Start the procedure using the OS/390 operator command START.
4. Use the OS/390 operator command STOP (P) to stop it again.

   You may alternatively use the operator command MODIFY (F): `MODIFY <started-task-name>,EOJ`

   Note:
   If you terminate the server using the operator command CANCEL, the TCP/IP port numbers used by System Management Hub may not be freed. In this case, use the appropriate commands to TCP/IP to display (NETSTAT) and free (DROP) the port numbers.

Verifying the Installation

After installing Natural Manager and System Management Hub on the mainframe, verify the installation by

1. invoking the System Management Hub startup JCL; and
2. connecting to the following URL from a valid workstation browser:
   `http://<host>:<MILHTTP_PORT>/smh/login.htm`

You are presented with the System Management Hub login screen allowing you to log in and start working with the Natural Manager environment.

Note:
If you have System Management Hub version 2.1.2 or above installed on non-OS/390 nodes (a Windows NT or 2000 node, for example), you need to copy additional icons from the OS/390 node into the "htmllayer/images/Natural" subdirectory of the target node. For example, on Windows NT, you need to copy the icons into "C:\Program Files\Software AG\System Management Hub\htmllayer\images\natural". The additional icons are located on OS/390 in the NDVvrs.AZIP file under "Argus/htmllayer/images/natural". You can bring the file to the target system using FTP in binary mode and unzip it using standard ZIP tools.

File Naming Specification for OS/390

The runtime system allows file names to be mapped to specific file protocols; for example, native OS/390 file input/output. For this reason, the default configuration of your System Management Hub makes file name references to the native OS/390 file system explicitly and other files default to the portable file system (PFS).

File names are specified using a 'typical' open system structure with each node of the file name separated by forward slashes with a filetype designation. For example:`file://level1/level2/level3/level4/filename.filetype` -where 'file://' is a static label used to identify native OS/390 files.

While this appears as UNIX or Windows format, the only objective of the name is to uniquely identify the file. The various levels are interpreted to make a sensible file name for the file system where the file resides.

However, the syntax difference between OS/390 files can be subtle. For example:

File Name Specified	Refers to . . .
`file://level1/level2/level3/mbrname.filetype`	PDS member: `LEVEL1.LEVEL2.LEVEL3(mbrname);` **Note:** '.filetype' is ignored.
`file://level1/level2/level3/mbrname`	The same PDS member as above.
`file://level1/level2/level3/level4/`	Sequential (PS-physical sequential) dataset: `LEVEL1.LEVEL2.LEVEL3.LEVEL4`

Note that the 'file://' specification differentiates the files above from logical files; for example, `/level1/level2/level3/level4` in the portable file system (PFS).

File names may also be referred to by a DD name reference that is then defined in the execution JCL. The following is the construct used in this case: `DD:<logical name>(<file name>)`

The <logical name> is a one to eight character name that the program wishes to access; the <file name> is optional.

If the file name is provided, the file with which the <logical name> is associated must be a PDS. Then the member with the name <file name> in this dataset is opened.

If the file is not capable of containing multiple members or the member does not exist, any attempt to open the file fails.

Using the System Management Hub with Natural

The System Management Hub is Software AG's cross-product and cross-platform product management framework providing efficient and user-friendly product administration, performance tuning, error analysis and interfaces to external systems.

The System Management Hub's architecture permits plug-in integration of existing products, separation of implementation-specific management technologies and re-use as product-independent, cross-platform framework.

- Log-in Procedure
- SMH User Interface
- Selecting a Host
- Refreshing the Display
- Description of Functionality

Log-in Procedure

The System Management Hub runs in a browser. This can be either Microsoft Internet Explorer 4.01 or above, or Netscape Communicator 4.7.

You invoke the System Management Hub with the following URL:

`http://<machine name>:<system management hub port number>/smh/login.htm`

Under Windows, you can simply access the System Management Hub as described above. The correct URL is automatically provided in the shortcut.

To log in to your local installation of System Management Hub

1. Start your Web browser.
2. Enter the URL of your local installation (MIL server) in the address line of your browser.
   This displays an HTML page containing the following in your browser:
   (smh screenshot)
3. In the System Management Hub Welcome screen, type in your user name and your password and choose the **Login** button.

SMH User Interface

The user interface for the System Management Hub is divided into three parts:

- Navigation/Tree-View Frame (top left)
- Command Frame (bottom left)
- Display/Detail-View Frame (right)

Navigation/Tree-View Frame

Contains all hosts and products known to the System Management Hub. You can expand or collapse the tree structure by clicking the plus or minus sign in front of an object.
Alternatively, you can single-click on an object to select it - in this case, details about this object are displayed in

the detail-view frame, but the object is not expanded.
If you double-click on an expandable object, the object will be expanded and details about the object are displayed in the detail-view frame.

Command Frame

The content of the command frame depends on the object you select from the tree-view frame and on the current status of the object. Only those commands that are available for the selected object are displayed. If you select another object, the content of the command frame will be refreshed accordingly.

Display/Detail-View Frame

This frame is used to display details of selected objects or to accept user input.

Selecting a Host

To select a host (example: IBM mainframe)

1. Click the object in the tree-view frame.
2. In the Login to Host dialog, enter your user ID and password.
   If they are identical with those you entered when logging in to the System Maintenance Hub, you can leave these input fields blank.
3. Click the Login button.

Refreshing the Display

The **Refresh** button is always available. It is used to refresh the contents of the tree-view frame and the detail-view frame. Thus, you can check whether the status or information has changed.

Description of Functionality

- Overview of Installed Versions
- Overview of Natural System Files
- Overview of Natural Subproducts
- Monitoring and Administration of Natural Servers (in preparation)

Overview of Installed Versions

When you use the System Management Hub views in conjunction with Natural, all network-wide Natural installations (nuclei and system files) are displayed. For each installed Natural, the version information is shown.

When you expand a node of a specific Natural nucleus, all Natural subproducts and nucleus components (Natural add-on products, Natural Security, etc.) are shown.

Clicking a subproduct will display the current update information as IUPDs and ZAPs. This corresponds to the Natural system functions DUMP ZAPS that is available on mainframes.

Overview of Natural System Files

Displays all system files registered. The detail-view frame shows the Natural subproducts installed on the selected system file. Selecting a subproduct will display detailed installation data.

Choosing the Show History button will invoke a complete update history for the specific subproduct. This information corresponds to the Natural system function SYSPROD.

Overview of Natural Subproducts

Displays the same information that is available with the Natural system files, but sorted by Natural subproducts.

Monitoring and Administration of Natural Servers (in Preparation)

In the System Management Hub views, you can see all defined Natural servers. They are marked as active or inactive. In addition, you can start or stop a server from within that view.

Note: This function is currently not yet available.

SPoD-Specific Limitations and Considerations

When you are working with Natural Single Point of Development, you will encounter a few limitations which are due to the different capabilities of the graphical user interface available on the local site and the character-based user interface that exists on the remote site. Also, some restrictions exist which will be eliminated in one of the next releases. In addition, this document includes hints which are important for the efficient use of the remote development facilities.

The following topics are covered:

- Limitations
- Performance Considerations

Limitations

- Execution of Programs Calling CICS-Related 3GL Programs
- Execution of Programs Accessing DL/I Databases
- PC Down/Uploads Using Natural Connection
- System Commands
- Moving/Copying Error Messages
- MOVE, COPY under Control of Natural Security
- LIST DDM, EDIT DDM
- Classes in Tree View
- Maps Containing GUI Elements
- Field Sensitive Maps
- Resources
- Dialogs
- Natural ISPF Macros and Recordings
- SYSLIB/SYSLIBS
- Allow Lower Case Input in Program Editor of Natural Studio
- Session Parameter NC
- Terminal Emulation
- Dependencies between XRef GUI Client and Predict
- Remote Debugging
- Execution of Programs Accessing DB2 Databases

Execution of Programs Calling CICS-Related 3GL Programs

The execution of programs calling 3GL programs which in turn use CICS-specific information or issue CICS-specific calls (CICS EXEC ...) is not possible with this version of Natural Single Point of Development, but is planned for a future version.

Execution of Programs Accessing DL/I Databases

The execution of programs accessing DL/I databases is not possible with this version of Natural Single Point of Development, but is planned for a future version.

PC Down/Uploads Using Natural Connection

The execution of programs which use Natural Connection to perform a PC down/upload is not possible with this version of Natural Single Point of Development. It is planned for a future version.

System Commands

- System Command DELETE, RENAME
- System Command SYSDDM
- System Commands Unavailable for Remote Development
- System Commands Entered Directly on the Development Server

System Commands DELETE, RENAME

These system commands are not available with this version, but are planned for a future version. Instead of DELETE, you can use the system commands UNCAT, PURGE and SCRATCH and the Delete dialog from the context menu. Instead of RENAME, you can use the rename dialog from the context menu.

System Command SYSDDM

The system command SYSDDM is not available, since the DDMs are listed in the tree view under the node DDM, and because all functions of the utility SYSDDM are available by using the context menu or the menu bar.

System Commands Unavailable for Remote Development

The following system commands are not available, since their use would make no sense with a graphical user interface:

- EDT
- HELLO
- MAINMENU

System Commands Entered Directly on the Development Server

All system commands which are entered not in the user interface of Natural Studio but directly on the server, via the stack by using the Natural session parameter STACK or in a program by using the STACK TOP COMMAND or by entry in the terminal emulation window, are executed directly by the Development Server without control of Natural Studio. As a result, the character-based representation of the corresponding command appears in the terminal emulation window. For example, when you specify the session parameter STACK=(LOGON XYZ;L * *) when mapping a server, the result will be that the system command LIST is output in the terminal emulation window.

It is even possible to invoke the mainframe editors. However, this may lead to inconsistencies (see also Object Locking). Therefore, you are strongly recommended to use only the GUI editors.

For the next version of of Natural Single Point of Development, it is planned that all system commands for which a graphical presentation exists in Natural Studio will be passed by the Development Server to Natural Studio for execution in itsgraphical user interface.

The commands HELLO and MAINMENU do not cause a screen output on the development server side, since this would not make any sense in the SPoD environment. The menu-driven user interface is that of Natural Studio.

Moving/Copying Error Messages

Moving and copying of error messages is different in remote and local environments:

- When error messages are moved or copied within the remote environment or are moved or copied from the local to the remote environment or vice versa:
  the error messages involved are merged, that is,
  - error messages which already exist in the target environment are replaced,
  - messages which do not exist in the source library are kept in the target library,
  - messages which do not exist in the target library are added.
- When error messages are moved or copied within the local environment, the messages involved are handled on file level, that is,
  - all error messages (that is, files) of a language are deleted and
  - the file from the source library is created anew in the target library.

MOVE, COPY under Control of Natural Security

Although these functions are executed using the GUI functions Cut, Copy, Paste, Drag and Drop, their use in the remote development environment can be restricted only via the security profile of the mainframe Natural utility SYSMAIN.

LIST DDM, EDIT DDM

In contrast with a pure Natural mainframe environment, that is, without remote development from Natural Studio, the command EDIT DDM is available also from a user library. This means that it is not necessary to expand the DDM node in the tree view to be able to edit a specific DDM. However, when Natural Security is used, the use of the commands LIST DDM and EDIT DDM can be restricted only via the security profile of the mainframe Natural utility SYSDDM.

Classes in Tree View

Although classes are shown in the tree view, the command OPEN is available only in the file view display. Performing an OPEN command on a class in the file view will invoke the program editor. Similarly, new classes can be created only in the file view, that is, by first creating an object of type Program which then can be saved as an object of type Class.

With one of the next versions of Natural Single Point of Development, it will be possible to create/edit the classes in all types of views using the Class Browser.

Maps Containing GUI Elements

Maps containing GUI elements can be moved or copied from the local environment to a remote environment. However, the GUI elements are not displayed when the map is being tested or executed on the remote environment.

Note:
Some GUI elements (push buttons, menus and bitmaps) of a map are discarded when a SAVE or STOW system command is executed for the map in a mainframe environment.

Field Sensitive Maps

For these maps the consistency check for a map field is made as soon as the data field is filled out by the user. Field Sensitive Maps can be moved or copied from the local environment to a remote environment. However, a field sensitive map can not be tested or executed on a remote mainframe environment.

Resources

Since the object type Resource is unknown in a mainframe environment, it is not possible to copy or move objects of this type from the local environment to a remote environment. Combined operations, for example, simultaneous moving of resources and programs are not possible.

Dialogs

Dialogs can be stored on the mainframe. Therefore it is possible to move or copy dialogs from the local environment to a remote environment. Private Resource Files of a dialog will not be moved or copied together with the dialog. It is also possible to list dialogs in a remote environment. The creation of new dialogs or editing dialogs is not possible in a remote environment.

Natural ISPF Macros and Recordings

As the object types Natural ISPF Macro and Recording available with Natural for Mainframes cannot be processed by Natural Studio, they will not be displayed in the tree view of the library work space. If a library consists only of such object types, the library will be displayed nevertheless in the tree view, but without any subnodes.

If a library containing such object types is deleted, then the objects of these two specific object types will not be deleted and the library will continue to be displayed in the tree view.

Objects of the types Natural ISPF Macro and Natural ISPF Recording cannot be linked to an application.

SYSLIB/SYSLIBS

The restricted libraries SYSLIB/SYSLIBS of the server are not shown in Natural Studio’s tree view, because a logon to these libraries is not possible. These libraries can be modified only by using a Natural utility such as INPL, NATUNLD/NATLOAD or SYSMAIN.

Allow Lower Case Input in Program Editor of Natural Studio

The program Editor of Natural Studio is case-sensitive, that is, lower case input will be included in the program source in lower case. The compiler on the Development Server, however, expects upper case code in its normal setting. This issue can be fixed by setting the compiler option LOWSRCE=ON. But this setting will have specific side effects which should be noticed. Refer to the CMPO profile parameter in the Natural Reference documentation.

Session Parameter NC

With this version, it is not possible to modify the NC parameter setting for the Development Server using the Globals dialog.

Terminal Emulation

The terminal emulation supports 3270 Model 2 screens. The support of 3270 Model 3, 4 and 5 screens is planned for one of the next versions of Natural Single Point of Development.

Dependencies between XRef GUI Client and Predict

After having completed the installation of Natural Version 5.1.1 for Windows on the client side and the Natural Development Server (NDV) Version 1.1.1 on the server side, you will be able to do remote development without having an installed version of Predict. However, the following dependencies should be noticed:

- If Predict is not installed on the server, there will be the following limitation in the XRef GUI Client's functionality:
  If you are using dynamic language assigned in calling other object like `'INPUT USING MAP 'MAP1&'`, the connection between caller and called object cannot be retrieved by using the XRef GUI Client.
- If you have Predict installed, the possible languages and the position of the language sign need to be defined on the server in Predict.

Remote Debugging

When the remote debugging facility was implemented, the goal was not to provide any new functions, but to support the existing essential debugging functions under the Natural Development Server. These functions are:

- Stepmode
- Breakpoints
- Watchpoints
- Display and modification of variables and their contents during a break

Generally, it was intended to provide for compatibility between the debug functionality that exists in a Natural on mainframes and a Natural on PC environment. Hence, the current state of development constitutes the lowest possible common denominator. Especially, the debug statistics as supported on mainframe are not yet supported in a remote debug environment.

Which Differences Exist in Debugging on Mainframe and on PC?

The following is an overview of differences that exist between Natural debugging in a mainframe (MF) environment and debugging in a PC environment (PC). The remote debugging between PC and mainframe supports the functionality as pointed out in the cases below for mainframe (MF):

- **Restarting a Debug Process**

  MF The restart function is not supported.

  PC Debug on PC offers a special restart function which is not available for remote debugging on mainframe.

- **System Variables**

  MF System variables can be displayed, but not modified. It is not possible to set watchpoints for system variables.

  PC System variables can be modified. It is possible to set watchpoints for system variables.

- **Watchpoint for Array Elements**

  MF Watchpoints are supported for single array elements.

  PC It is possible to define a watchpoint for an entire array or a selected range.

- **Several Breaks (BP/WP) per Line Number of the Program**

  MF Multiple breaks or interrupts may arise for one and the same line number (because of multiple definitions of breakpoints or watchpoints).

  PC Stepmode, BP and WP settings together result in a maximum of one break per line number.

- **Breakpoints**

  MF Breakpoints can be defined for programs which are found in the current library or in any steplib.

  PC Debugging allows to define breakpoints for programs in any library (not necessarily current library or steplib).

- **BP-END**

  MF BP-END becomes active before the program is left. Normally, this is triggered by the END statement. However, if the END statement in the program code is preceded, e.g. by a FETCH statement, the BP-END condition will become effective already there.

  PC BP-END always refers to the END statement.

- **Different Handling of Watchpoints with Comparison Operators**

  MF The watchpoint becomes active when the watchpoint variable has changed and when the comparison condition is met.

  PC The watchpoint becomes active when the comparison condition for the watchpoint variable is met.

- **Leaving the Debugger**

  MF When you leave the Natural Debugger on mainframe, the program execution continues.

  PC Leaving the Debugger causes the program execution to be stopped.

- **Debugging of Programs which are Called through the Stack**

  MF Stacked programs can be debugged, when any breakpoint or watchpoint has been defined, but they cannot be entered automatically in stepmode..

  PC Programs can be entered automatically in stepmode.

Execution of Programs Accessing DB2 Databases

In the case of an access to DB2 from a program executed on the Natural Development Server, the user ID for the database access is not the client's user ID, but the job or task name of the development server's started task.

Performance Considerations

The working situation displayed in the library workspace of Natural Studio is based on the representation of the **entire** user system files. The tree view window opens when the user connects to the Natural Development Server. For this, the entire system file has to be analyzed and the corresponding information has to be transferred from the Natural Development Server to the Natural Studio Client. In the case of very large system files, the build-up of the tree view window can be very time consuming. A status information displayed in the status bar keeps the user informed about the progress of the screen build-up operation. This is to avoid the impression that the connection to the Natural Development Server might be interrupted.

Tip:	Switch on the status bar using the View - Status Bar function of the Menu bar.
	Make sure that the transfer rate of your network is 10 Mbit/s at minimum.

In the default configuration of Natural Studio, all operations which result in a modification of the system file, for example, moving or copying objects, but also a SAVE or STOW command, will cause the tree view window contents to be refreshed, which can be very time consuming in the case of very large system files.

Tip:	Disable the automatic refresh function by choosing **Tools** > **Options** > **Workspace** and deactivating the function "Perform automatic refresh".
	The automatic refresh may then be activated if actually needed, for example, on a specific library node or for all user libraries, by activating the refresh function in the context menu.

Since the tree view of the application workspace displays only the objects that are linked to the application, the build-up of its tree view screen is consequently considerably faster, which is another advantage of using the application workspace.

